POSTSCRIPT FOR TECHNICAL DRAWINGS

PSPLOT: A FORTRAN-CALLABLE
POSTSCRIPT PLOTTING LIBRARY

USER'S MANUAL

Kevin E. Kohler

Nova Southeastern University
Oceanographic Center
8000 North Ocean Drive
Dania, Florida 33004

1. INTRODUCTION

This manual is intended for users interested in generating two-dimensional technical drawings or graphics for technical journals in PostScript format. The manual describes a library of Fortran-callable subroutines which can be combined in a calling program to produce PostScript plot files. The purpose, syntax, and calling convention of each subroutine is presented , along with plotting examples.

PostScript is a very robust graphics and typesetting language with wide-ranging capabilities. Since the focus of the subroutines in this library is to produce technical drawings, many of the "artistic" features of PostScript have not been addressed or included here.

The name PostScript is a registered trademark of Adobe Systems Incorporated. All instances of the name PostScript in this manual are references to the PostScript language as defined by Adobe Systems Incorporated.

Send questions, comments, suggestions to *kevin@ocean.nova.edu*.

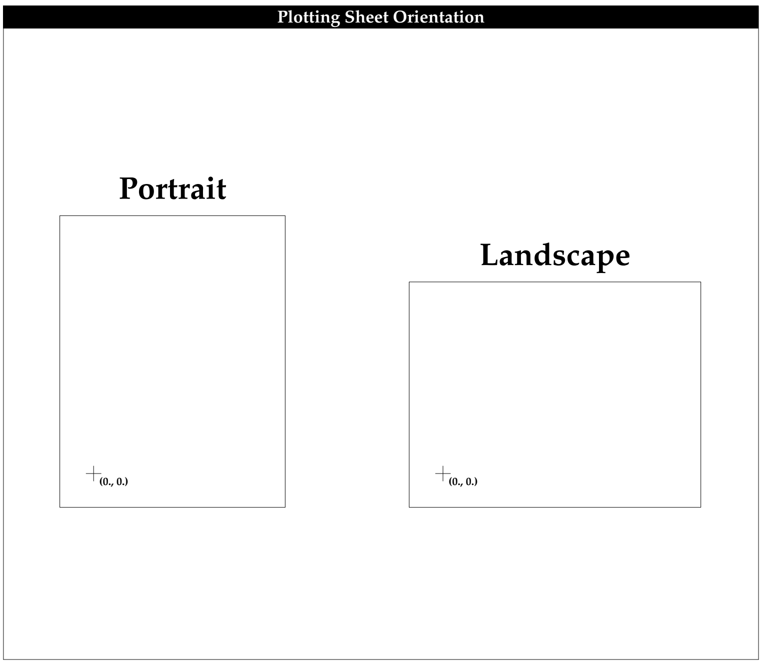
2. PLOTTING BASICS

This section will present the conventions used by the plotting subroutines in the PSPLOT plotting library . This information can be used to create customized plotting subroutines.

A *plotting session* is a set of plotting instructions in a user's application program which produces hardcopy graphic display output. The output can be either a single plot or graph or a set of graphs. **Every plotting session must begin with a call to subroutine PSINIT (with the exception of a call to NEWDEV) and end with a call to subroutine PLOTND.**

Pen movement is occasionally mentioned throughout this manual. While PostScript does not use an actual pen for graphic production, it is useful to visualize the plot commands as directing the movements of a pen of variable thickness to a specified (x,y) coordinate, with the pen being either up or down. The plotting subroutines are simple, user-callable commands which direct the movements of an imagined pen upon a plotting sheet.

The initial coordinate origin is approximately .5 inches from the bottom and left paper edges. The orientation of the page must be specified to be either **portrait** (short side horizontal) or **landscape** (long side horizontal) and is set by the call to PSINIT (which see). The figure below shows the paper orientation and beginning plot origin for portrait and landscape modes. The current plot origin can be relocated to other positions during the plotting session to provide new reference points for subsequent plotting commands. When the next graph is started, the new origin should be placed far enough away to avoid overlapping the just-completed graph.



All plotting commands use an *absolute* plot coordinate system. This means that all coordinates passed to plotting subroutines are expected to be in terms of distance from the current plot origin. This is contrast to a *relative* plot coordinate system, in which coordinates are assumed to be in terms of distance from the current pen position.

In all plotting subroutines, plot coordinates, character heights, distances, etc. are measured in *inches*.

Unless otherwise noted, any subroutine argument which specifies an angle is stated in *degrees relative to the X axis, with positive angles measured counterclockwise from the X axis*.

Most character variables are specified as Hollerith, rather than character strings. The reason for this is primarily historical, combined with the fact that the enormous amount of code already written and in use at my site precluded a comprehensive revision. This should not cause any problems, however.

The initial font of a plot session is Helvetica, with a size of 12 points. This can be changed permanently in subroutine PSINIT. Of course, the current font can always be changed with subroutine SETFNT. PSPLOT supports the standard 35 fonts found on most PostScript printers.

PSPLOT supports color, although all examples in this manual are shown in grayscale for the purposes of reproduction. Color is specified in the relevant subroutines as red, green, and blue (RGB) values.

Continuation allows you to append a character string or number to the end of a previously plotted string or number. The coordinates of the appended string are automatically calculated. Whether or not a subroutine supports continuation is stated in subroutine description in the next section.

Continuation is specified by setting the X and/or Y coordinate argument in calls to the subroutines listed above to 999, and may be applied to X and Y coordinates separately. A subroutine call with continuation must immediately follow the previous plotting call. Continuation is useful when plotting strings containing variable values, such that the resultant string length is not known beforehand.

For example,

```
ND=10
CALL KEKNUM (1., 2., .15, FLOAT(ND), 0., -1, 0)
CALL KEKSYM (999., 999., .15, 5H DAYS, 0. ,5, 0)
```

produces "10 DAYS".

3. USER-LEVEL PLOTTING ROUTINES

This section describes the subroutines in the PSPLOT plotting library at the time of this writing. These routines are Fortran-callable from an application program. Below is a brief summary of the subroutines, followed by an alphabetical listing of each subroutine containing a more detailed description of its function, syntax, and calling arguments.

ARC	Draws an arc of a specified radius and center.
AROHED	Draws an arrowhead at a specified location.
ARROW	Draws an arrow including the shaft.
AXIS	Draws an axis of a graph, with user-specified axis titles.
BLKSTP	Strips out embedded blanks in a character string.
BORDER	Draws a rectangular border with tick marks.
CHOPIT	Logically closes the current page and begins a new one.
CIRCLE	Draws a circle with specified center, radius and fill.
CLIP	Inserts the PostScript command *clip* into the output file.
CLIPBOX	Defines the current clipping area.
COLBOX	Fills regions with specified color.
CONCOLR	Draws colored contour plots of a 2-dimensional array of regularly-spaced data.
CONFILL	Draws grayscale contour plots of a 2-dimensional array of regularly-spaced data.
CONREC	Draws contour plots of a 2-dimensional array of regularly-spaced data.
CURVE	Draws a curve between 2 points with specified beginning and ending slopes.
DRWCRV	Connects data points with a solid line.
DRWTRI	Draws a triangle.
DSHCRV	Connects data points with a dashed curve.
DSHLIN	Draws a dashed line between two points.

FACTOR	Enlarges or reduces a plot by a specified ratio.
FAROHED	Draws a "fancy" arrowhead.
FILLBOX	Fills a shape with a specified grayscale level.
FILRGN	Same as FILLBOX, except it restores the current gray level after filling.
FILRGNC	Fills a shape with the current gray level or color.
GREST	Calls the PostScript operator *grestore*.
GRKSYM	Plots a Greek symbol.
GSAV	Calls the PostScript operator *gsave*.
HILITEC	Prints text surrounded by a rectangular box (color).
HILITEG	Prints text surrounded by a rectangular box (grayscale).
INTEGRAL	Plots an integral with upper and lower limits.
KEKEXP	Plots a floating point number in exponential format.
KEKFLT	Plots a floating point number in floating point format.
KEKNUM	Plots a floating point number.
KEKSYM	Plots a character string.
KEKSYMC	Plots a character string, with the text string being chararacter rather than Hollerith.
KEKSYMO	Plots a character string with the characters outlined instead of filled.
LENSTR	Computes the number of characters in a character string.
NEWDEV	Specifies the name of the output PostScript file.
NUMBER	Plots a floating point number. Similar to KEKNUM, but justification and continuation are not supported.
ONEHLF	Draws the symbol ½.
OVERBAR	Draws a character string with an overbar.
OVRSBSP	Draws an overbar over a subscripted and superscripted character string.

OVRSBSPG	Draws an overbar over a subscripted and superscripted Greek symbol.
OVERSUB	Draws an overbar over a subscripted character string.
OVERSUBG	Draws an overbar over a subscripted Greek symbol.
OVERSUP	Draws an overbar over a superscripted character string.
OVERSUPG	Draws an overbar over a superscripted Greek symbol.
OVRGRK	Draws an overbar over a Greek symbol.
PLOT	Directs "pen" movement to a specified (x,y) point with the "pen" either up or down.
PLOTND	Closes the output PostScript file and terminates the plotting session.
PLSMIN	Draws the ± character.
PRIME	Draws the ' character.
PSINIT	Called to begin a plotting session. It also specifies portrait or landscape mode.
RECT	Draws a rectangle.
RECTFILC	Draws a rectangle and fills it with specified red, green, and blue color levels.
RECTFILG	Draws a rectangle and fills it with specified gray level.
ROTATE	Rotates the current coordinate system by a specified angle. Essentially, ROTATE issues the PostScript *rotate* command.
RRECT	Draws a rectangle with rounded corners. It can then fill the rectangle with the current graylevel or RGB values.
SETCOLR	Sets the current color values.
SETFNT	Sets the current font.
SETGRY	Sets the current gray level value.
SETLW	Sets the current line width.
SIGMA	Draws the symbol Σ, with upper and lower limits.

SLDCRV	Connects data points with a solid curve.
SLDLIN	Draws a solid line between two points.
SQRSGN	Draws the radical (square root) sign.
SQUARE	Draws a square.
STROKE	Calls the PostScript operator *stroke*.
SUBBER	Draws a subscript.
SUBBERSP	Draws a subscript of "special" characters.
SUBSUP	Draws a subscript and superscript.
SUBSUPSP	Draws subscripts and superscripts of "special" characters.
SUPER	Draws a superscript.
SUPERSP	Draws a superscript of "special" characters.
SYMBOL	Plots a character string. Similar to KEKSYM, but justification and continuation are not supported.

The plotting subroutines in the PSPLOT plotting library are written in Fortran and are called by user-written application programs. In the following subroutine descriptions, integer variables are denoted by arguments beginning with the letters I-N, as per Fortran convention. All other variables are real (4 byte) floating point Fortran and are called by user-written application programs. All of the subroutines are written in single precision. Hence, if your program generates data in double precision, any arguments sent to the plotting library subroutines must first be converted to single precision.

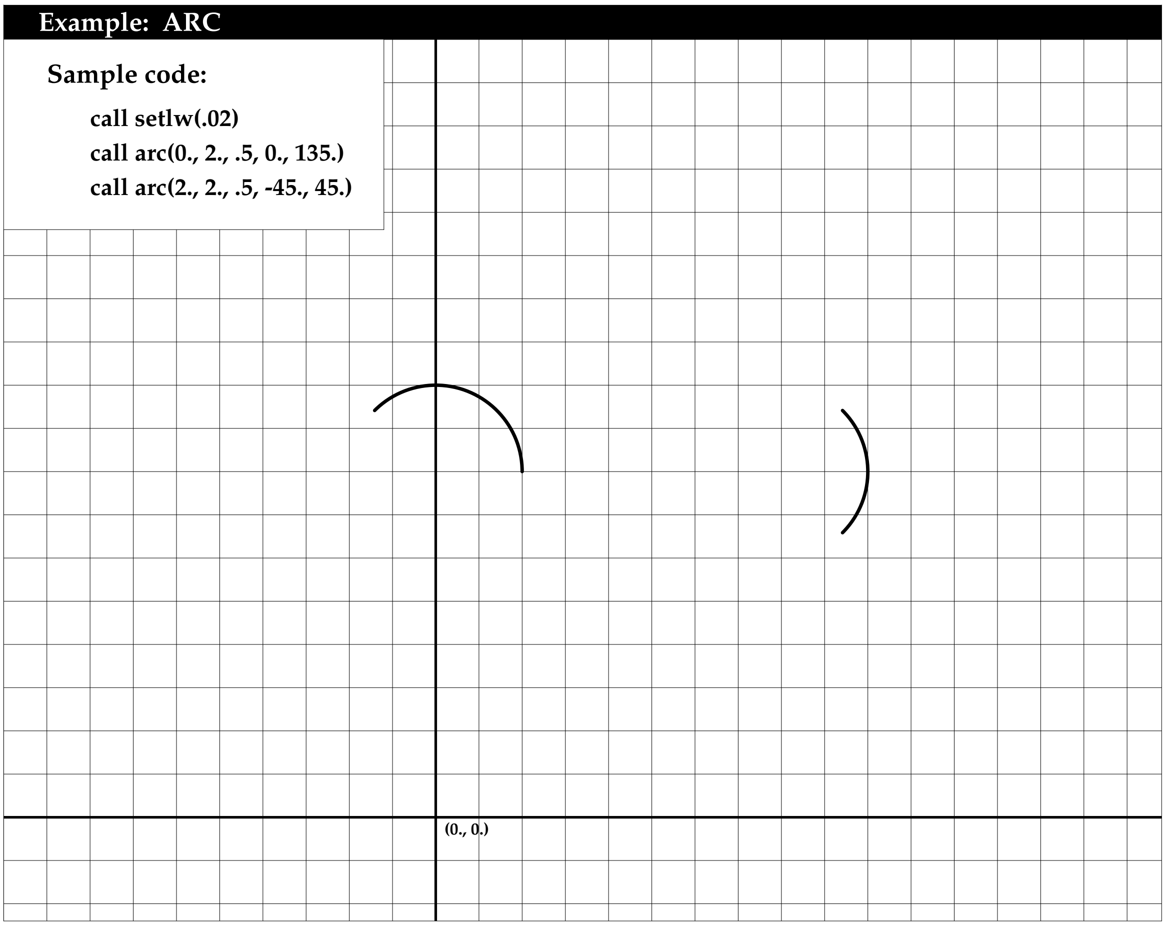
ARC

PURPOSE **ARC** draws an arc having a specified radius and arc center.

SYNTAX CALL ARC (XC, YC, RAD, ANG1, ANG2)

<u>*ARGUMENTS*</u>

XC, YC	X,Y coordinates of the center of the arc.
RAD	Radius of arc.
ANG1	Angle of arc beginning.
ANG2	Angle of arc ending.



AROHED

PURPOSE **AROHED** draws an arrowhead at a specified location.

SYNTAX CALL AROHED (XPP, YPP, DIR, AROLNP, SPRANG, LOCXY)

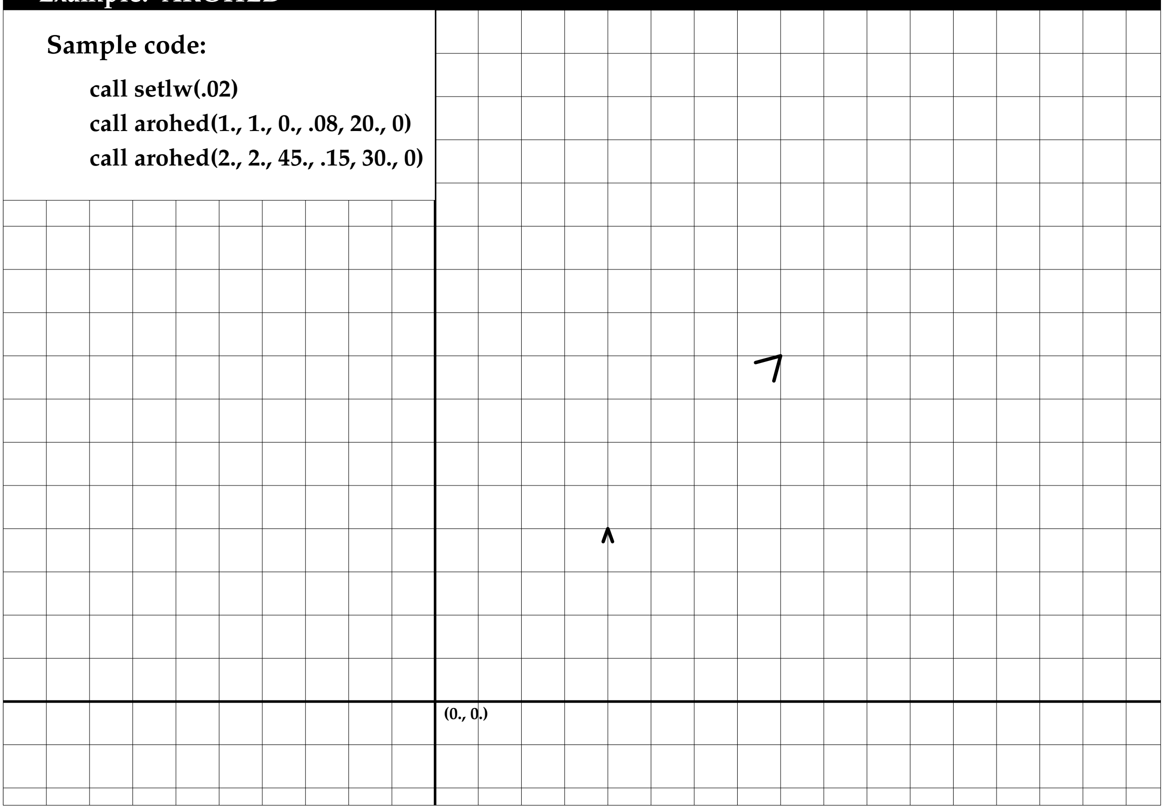
ARGUMENTS

- XPP,YPP — X,Y coordinates of the tip of the arrowhead.
- DIR — Direction of arrowhead, measured east from north.
- AROLNP — Length of arrowhead sides.
- SPRANG — Half the angular spread of arrowhead.
- LOCXY — Location of arrowhead point
  - LOCXY=1 — XPP,YPP at arrowhead point (most commonly used)
  - LOCXY=2 — XPP, YPP at center of arrowhead.
  - LOCXY=3 — XPP,YPP at tail of arrowhead.

Example: AROHED

Sample code:

```
call setlw(.02)
call arohed(1., 1., 0., .08, 20., 0)
call arohed(2., 2., 45., .15, 30., 0)
```



ARROW

PURPOSE **ARROW** draws an arrow, with vector length, at a specified location.

SYNTAX CALL ARROW (XSS, YSS, XPP, YPP, AROLNP, SPRANG, LOCXY)

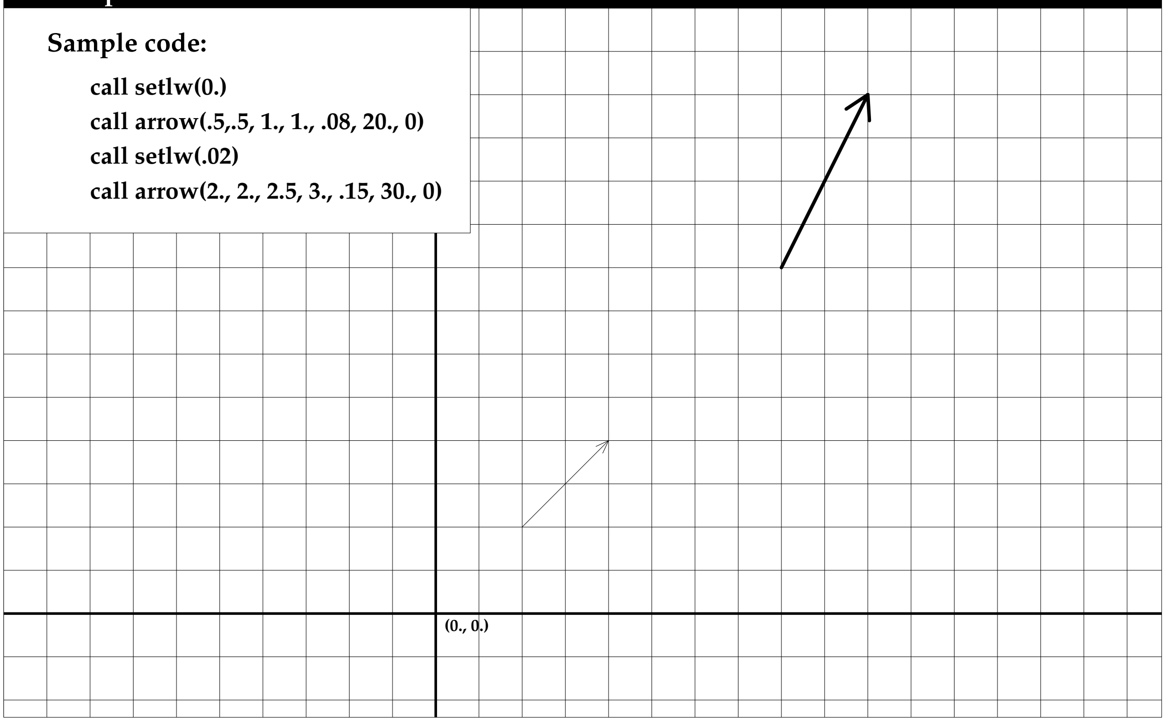
<u>*ARGUMENTS*</u>

XSS,YSS	X,Y coordinates of the vector length origin.
XPP,YPP	X,Y coordinates of the tip of the arrowhead.
AROLNP	Length of arrowhead sides.
SPRANG	Half the angular spread of arrowhead.
LOCXY	Location of arrowhead point LOCXY=1 XPP,YPP at arrowhead point (most commonly used) LOCXY=2 XPP, YPP at center of arrowhead. LOCXY=3 XPP,YPP at tail of arrowhead.

Example: ARROW

Sample code:

```
call setlw(0.)
call arrow(.5,.5, 1., 1., .08, 20., 0)
call setlw(.02)
call arrow(2., 2., 2.5, 3., .15, 30., 0)
```



AXIS

PURPOSE **AXIS** draws an axis of a graph, with user-specified axis titles. Tick marks are drawn with a spacing of one inch.

SYNTAX CALL AXIS (XPP, YPP, IBCD, NC, AXLEN, THETA, RMIN, DEL)

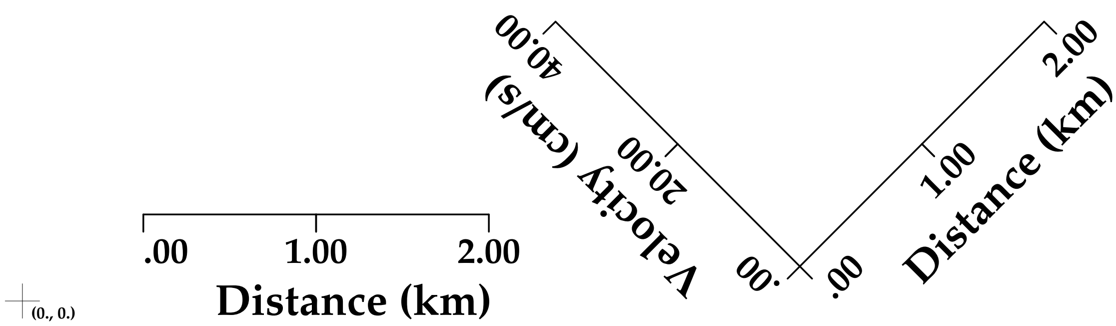
ARGUMENTS

XPP,YPP	X,Y coordinates of axis origin.
IBCD	Axis title (Hollerith).
NC	Number of characters in title. Its sign is used to specify on which side of the axis the title is to appear: positive for the counterclockwise side of the axis, negative for the clockwise side. Positive labeling is generally used for Y axes, negative for X axes.
AXLEN	Length of axis.
THETA	Angle of axis. Generally, 0 for X axes, 90 for Y axes.
RMIN	The starting value and annotation of the first tick mark.
DEL	The data value increment between tick marks.

Example: AXIS

Sample code:

```
xlen=2.
ylen=2.
call setlw(.01)
call axis(.7, .5, 13HDistance (km), -13, xlen , 0., 0., 1.)
call axis(4.5, .2, 13HDistance (km), -13, xlen , 45., 0., 1.)
call axis(4.5, .2, 15HVelocity (cm/s), 15, ylen, 135., 0., 20.)
```



BLKSTP

PURPOSE **BLKSTP** strips out embedded blanks in a character string. It returns the new character string and the number of characters in the new string.

SYNTAX CALL BLKSTP (STR1, NDIM, STR2, NC)

ARGUMENTS

STR1	Input character string.
NDIM	Maximum length of STR1 as stated in calling program.
STR2	Output character string with embedded blanks removed.
NC	Number of characters in STR2.

BORDER

PURPOSE **BORDER** draws a rectangular border with tick marks. The lower left corner of the border is located at the current plot origin.

SYNTAX CALL BORDER(XLEN, YLEN, ITIC, IBRD, MAJX, MINX, MAJY, MINY)

<u>*ARGUMENTS*</u>

XLEN Length of x-direction side of border.

YLEN Length of y-direction side of border.

ITIC Four digit number which determines which of the border sides will contain tick marks. Each of the digits is either 0 or 1, and if set to 1, that side will contain tick marks. The border sides are ordered as follows:

Left-vertical Bottom Right-vertical Top

For example, if ITIC = 1011, all sides except the bottom would contain tick marks. Additionally, if ITIC<0, the tick marks will be drawn on the outside of the border than on the inside (default).

IBRD IBRD is similar to ITIC except that it determines which sides of the border will be drawn. Hence, you can have tick marks with no border, and vice versa.

MAJX Number of major divisions in the x-direction. A longer tick mark is drawn for these divisions.

MINX Number of minor divisions in the x-direction, i.e. the number of divisions per major division. A shorter tick mark is drawn for these divisions.

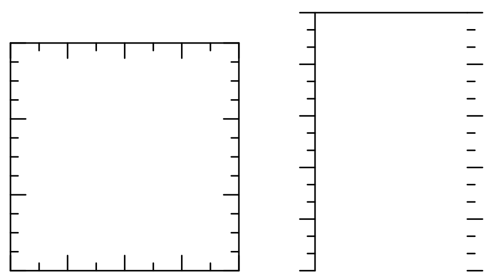
MAJY Same as MAJX in y-direction.

MINY Same as MINX in y-direction.

Example: BORDER

Sample code:

```
call setlw(.01)
call border(1.5, 1.5, 1111, 1111, 4, 2, 3, 4)
call plot(2., 0., -3)
call border(1., 1.7, -1010, 1001, 4, 1, 5, 3)
```



CHOPIT

PURPOSE **CHOPIT** logically closes the current graphics page and begins a new one.

SYNTAX CALL CHOPIT (XPP, YPP)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the initial plot origin of the next plot. This allows you to begin all plots at the same origin on each page.

Note: XPP and YPP are coordinates independent of the current scaling factor. That is, XPP and YPP represent actual inches, not scaled coordinates. The current scaling factor is reinstated after the new coordinate origin is set.

Subroutine CHOPIT should not be confused with subroutine PLOTND, which is called once at the end of each plotting session and which closes the entire output PostScript file.

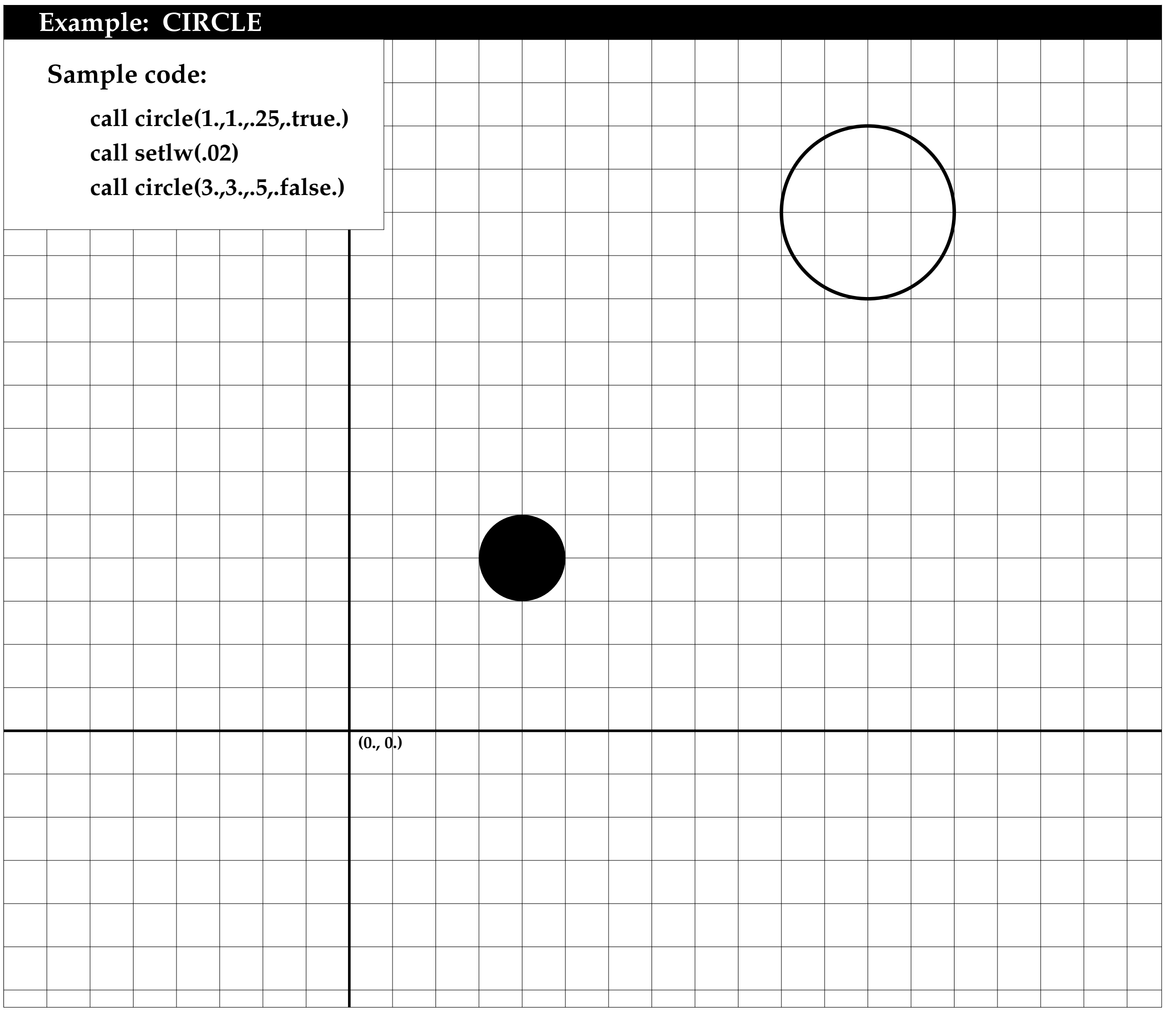
CIRCLE

PURPOSE **CIRCLE** draws a circle.

SYNTAX CALL CIRCLE (XC, YC, RAD, FILL)

ARGUMENTS

XC,YC	X,Y coordinates of the center of the circle.
RAD	Radius of the circle.
FILL	Logical value. If FILL=.TRUE., circle is filled with the current color or gray scale; otherwise it is outlined.



CLIP

PURPOSE **CLIP** inserts the PostScript command *clip* into the output file.

SYNTAX CALL CLIP

ARGUMENTS

NONE.

Note: Most uses of this command are bracketed by calls to GSAV and GREST.

CLIPBOX

PURPOSE **CLIPBOX** defines a clipping region.

SYNTAX CALL CLIPBOX (XPTS, YPTS, NPTS)

ARGUMENTS

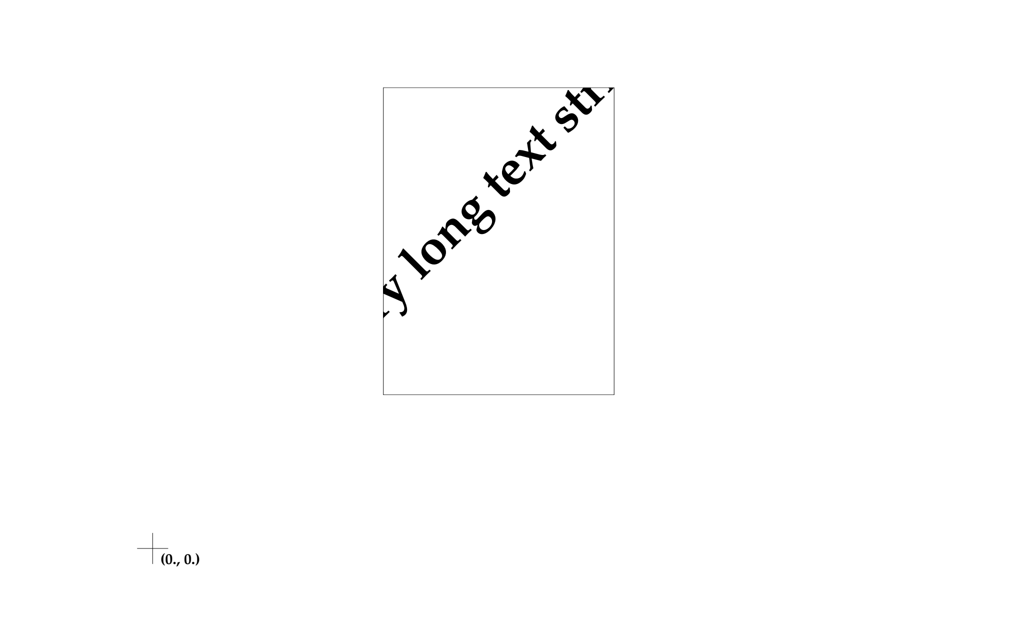
XPTS	Array containing x-coordinates of region path.
YPTS	Array containing y-coordinates of region path.
NPTS	Number of points in region path.

Example: CLIPBOX

Sample code:

```
dimension xa(4), ya(4)
data xa/1.5,3.,3.,1.5/
data ya/1.,1.,3.,3./
call clipbox(xa, ya, 4)
call keksym(.5, .5, .2, 23hA very long text string, 45., 23,0)
```

Code to outline clipping region not shown.



COLBOX

PURPOSE **COLBOX** fills a region with a specified color.

SYNTAX CALL COLBOX (XPTS, YPTS, NPTS, RED, GREEN, BLUE)

ARGUMENTS

XPTS	Array containing x-coordinates of region path.
YPTS	Array containing y-coordinates of region path.
NPTS	Number of points in region path.
RED	Red value of specified color in RGB color model.
GREEN	Green value of specified color in RGB color model.
BLUE	Blue value of specified color in RGB color model.

Arguments RED, GREEN, and BLUE must lie between 0. and 1.0, inclusive.

CONCOLR

PURPOSE **CONCOLR** draws color contour plots of a two-dimensional array of regularly spaced data.

SYNTAX CALL CONCOLR (ARR, IMAX, IEXT, JEXT, XLEN, YLEN, CVAL, COLOR, NVAL, IOFFP, SPVAL)

<u>*ARGUMENTS*</u>

ARR	Two-dimensional array containing regularly spaced data to be contoured. CONCOLR assumes ARR(1,1) is located at the lower left corner of the plot. Data points ARR(1,1) through ARR(IEXT,JEXT) are contoured in an area XLEN x YLEN.
IMAX	The first dimension of ARR in the calling program.
IEXT	Number of points in x-direction of ARR to be contoured.
JEXT	Number of points in y-direction of ARR to be contoured.
XLEN	X-direction length of plotting area.
YLEN	Y-direction length of plotting area.
CVAL	Array containing the values to be used for contour intervals (1 to NVAL). CVAL must be dimensioned with a dimension of at least 1. Unlike in subroutine CONREC, NVAL cannot equal 0.
COLOR	Array dimensioned (3,NVAL) containing the red, green, and blue values for each of the contour levels. The red value is stored in (1,n), the green value in (2,n) and the blue value in (3,n), where n specifies the contour index corresponding to CVAL. Regions less than or equal to the corresponding contour value CVAL are filled with the corresponding RBG values.
NVAL	Number of contour intervals. NVAL must be less than or equal to 100, and unlike in CONREC, NVAL cannot equal 0.
IOFFP	Flag indicating that grid boxes whose vertices have the value SPVAL are to be ignored during contouring.
SPVAL	Special value denoting which grid boxes are to be ignored during contouring.

See coding example for CONFILL.

CONFILL

PURPOSE **CONFILL** draws grayscale contour plots of a two-dimensional array of regularly spaced data.

SYNTAX CALL CONFILL(ARR, IMAX, IEXT, JEXT, XLEN, YLEN, CVAL, GRYLEV, NVAL, IOFFP, SPVAL)

ARGUMENTS

ARR Two-dimensional array containing regularly spaced data to be contoured. CONFILL assumes ARR(1,1) is located at the lower left corner of the plot. Data points ARR(1,1) through ARR(IEXT,JEXT) are contoured in an area XLEN x YLEN.

IMAX The first dimension of ARR in the calling program.

IEXT Number of points in x-direction of ARR to be contoured.

JEXT Number of points in y-direction of ARR to be contoured.

XLEN X-direction length of plotting area.

YLEN Y-direction length of plotting area.

CVAL Array containing the values to be used for contour intervals (1 to NVAL). CVAL must be dimensioned with a dimension of at least 1. Unlike subroutine CONREC, NVAL cannot equal 0.

GRYLEV Array dimensioned (NVAL) containing the grayscale values for each of the contour levels. Regions less than or equal to the corresponding contour value CVAL are filled with the corresponding grayscale values. Grayscale values range from 0. (black) to 1.0 (white).

NVAL Number of contour intervals. NVAL must be less than or equal to 100, and unlike in CONREC, NVAL cannot equal 0.

IOFFP Flag indicating that grid boxes whose vertices have the value SPVAL are to be ignored during contouring.

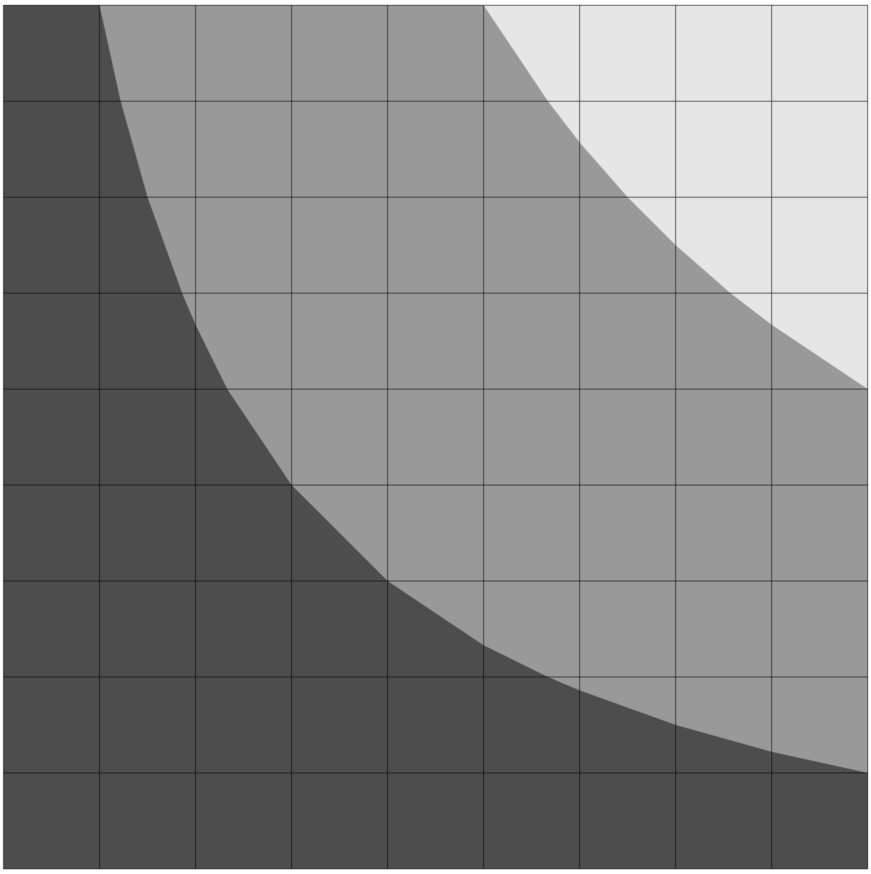
SPVAL Special value denoting which grid boxes are to be ignored during contouring.

Example: CONFILL

Sample code:

```
      do 10 j=1,10
      do 10 i=1,10
10    f(i,j)=i*j
      xlen=5.
      ylen=5.
      nval=6
      do 20 n=1,nval
      cval(n)=(n-1)*20
      if(cval(n).le.20.) then
       grylev(n)=.3
      elseif(cval(n).le.60.) then
       grylev(n)=.6
      else
       grylev(n)=.9
      endif
      continue
20    call confill(f,10,10,10,xlen,ylen,cval,grylev,nval,ioffp,spval)
```

Code to draw grid boxes not shown.



CONREC

PURPOSE **CONREC** draws contour plots of a two-dimensional array of regularly spaced data.

SYNTAX CALL CONREC(ARR, IMAX, IEXT, JEXT, XLEN, YLEN, CVAL, NVAL)

<u>*ARGUMENTS*</u>

ARR	Two-dimensional array containing regularly spaced data to be contoured. CONREC assumes ARR(1,1) is located at the lower left corner of the plot. Data points ARR(1,1) through ARR(IEXT,JEXT) are contoured in an area XLEN x YLEN.
IMAX	The first dimension of ARR in the calling program.
IEXT	Number of points in x-direction of ARR to be contoured.
JEXT	Number of points in y-direction of ARR to be contoured.
XLEN	X-direction length of plotting area.
YLEN	Y-direction length of plotting area.
CVAL	Array containing the values to be used for contour intervals (1 to NVAL). CVAL must be dimensioned with a dimension of at least 1. If NVAL = 0, the individual contour values are automatically computed.
NVAL	Number of contour intervals. If NVAL = 0, the number of contour intervals and the contour interval values are automatically calculated. This is helpful if the range of data values is not known beforehand. NVAL must be less than or equal to 100.

***<u>Special features of CONREC</u>*:**

If NVAL **<** 0, only high and lows are plotted instead of contours.

Common block CONPAR is used to control various characteristics of the contour plot from the calling program. The variables in CONPAR are shown below, along with their default values:

COMMON/CONPAR/ ISPEC, IOFFP, SPVAL, ILEGG, ILABB, NHII, NDECCN, NLBLL, LSCAL, LDASH, HGTLAB

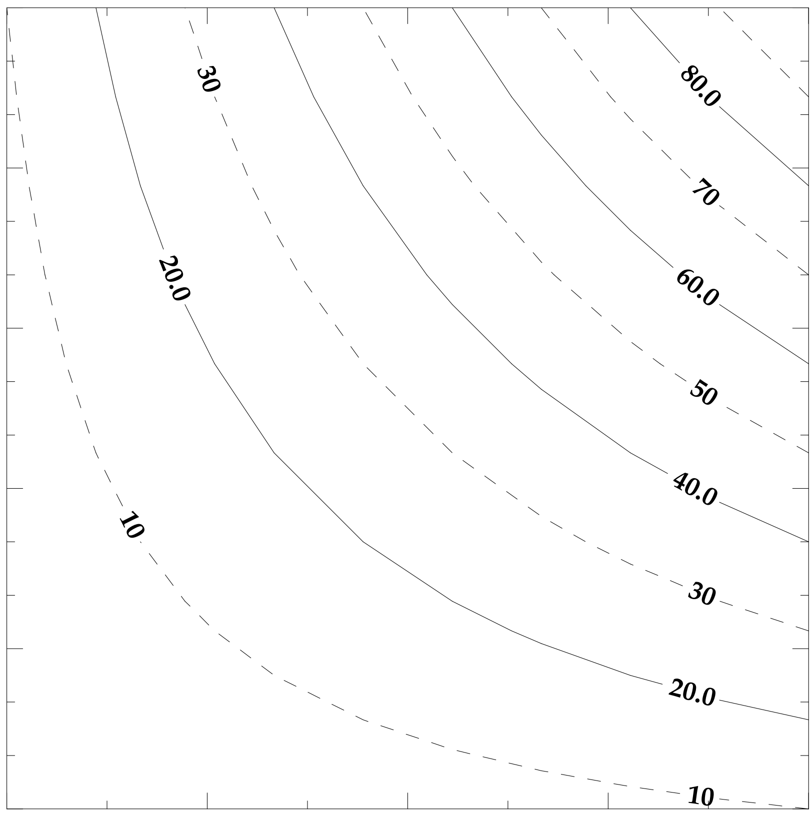
ISPEC If 0, variables in CONPAR have no effect. Default = 0.

IOFFP	If 1, you can specify regions not to contour by using SPVAL. Grid boxes with any corner values equal to SPVAL are not contoured. Default = 0.
SPVAL	Special value to denote areas not to contour (usually set to 999.). Default = 0.
ILEGG	If 0, legend below contour plot showing contour values, scale factors, etc. will not be printed. Default = 1.
ILABB	If 0, contour labels are not plotted. Default = 1.
NHII	If 0, H and L mark relative highs and lows, with the data value plotted underneath the symbol. If > 0, values are plotted at each array point. If < 0, neither of the above is done. Default = 0.
NDECCN	Number of digits to the right of the decimal point in the contour labels. Default = 1.
NLBLL	Number of unlabeled contours between labelled contours. Default = 3.
LSCAL	If LSCAL = 0, the contours are scaled such that 0 < labeled values < 100. If LSCAL = 1, the contours are not scaled, i.e. the contour values are the actual data values. Default = 0.
LDASH	Specifies whether contours are solid (LDASH = 0) or dashed (LDASH NE 0). If LDASH is not equal to 0, then LDASH specifies the type of dash line to use to draw the contours. The numeric value of LDASH is the number of times the dashed pattern (solid line/blank space) is repeated per inch. Default= 0.
HGTLAB	Specifies the height of contour labels. If HGTLAB=0, the height of the contours is .11 inches.

Example: CONREC

Sample code:

```
      do 10 j=1,10
      do 10 i=1,10
10    f(i,j)=i*j
      xlen=5.
      ylen=5.
      nval=6
      call border(xlen, ylen, 1111, 1111, 4, 2, 5, 3)
      do 20 n=1,nval
      cval(n)=(n-1)*20
20   continue
     nlbll=0
     lscal=1
     call conrec(f, 10, 10, 10, xlen, ylen, cval, nval)
     ldash=6
     ndecn=-1
     do 25 n=1,nval
     cval(n)=10.+(n-1)*20
25   continue
     call conrec(f, 10, 10, 10, xlen, ylen, cval, nval)
```



CURVE

PURPOSE **CURVE** draws a curve between two points, given the beginning and ending curve slopes.

SYNTAX CALL CURVE (X1, Y1, X2, Y2, X3, Y3, X4, Y4, CONTIN)

ARGUMENTS

X1,Y1 X,Y coordinates of the beginning of the curve.

X2,Y2 X, Y coordinates specifying the beginning curve slope. The initial slope of the curve is given by (Y2-Y1)/(X2-X1).

X3,Y3 X,Y coordinates specifying the ending curve slope. The end slope of the curve is given by (Y4-Y3)/(X4-X3).

X4,Y4 X,Y coordinates of the endpoint of the curve.

CONTIN Logical value. If CONTIN=.false. then the curve drawn starts a newpath; otherwise, the curve is appended to the current path.

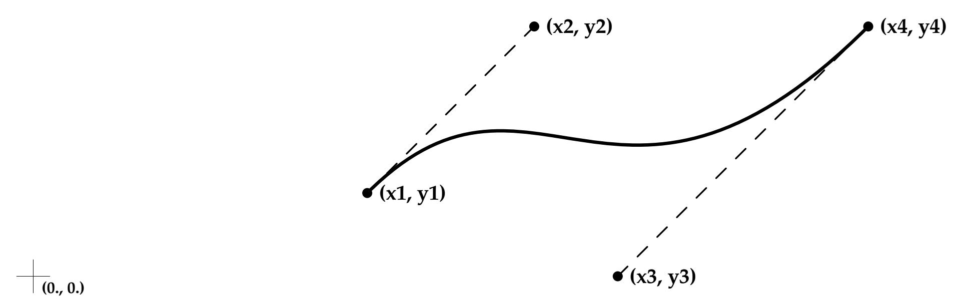
Note: The curve produced by CURVE **is not** automatically stroked.

Example: CURVE

Sample code:

```
data x1, x2, x3, x4/2., 3., 3.5, 5./
data y1, y2, y3, y4/.5, 1.5, 0., 1.5/
call setlw(.02)
call curve(x1,y1,x2,y2,x3,y3,x4,y4,.false.)
call stroke
```

Code to indicate points and dashed lines not shown.



DRWCRV

PURPOSE **DRWCRV** draws a solid curve.

SYNTAX CALL DRWCRV (XARR, YARR, NPTS, THK, CLOSER)

ARGUMENTS

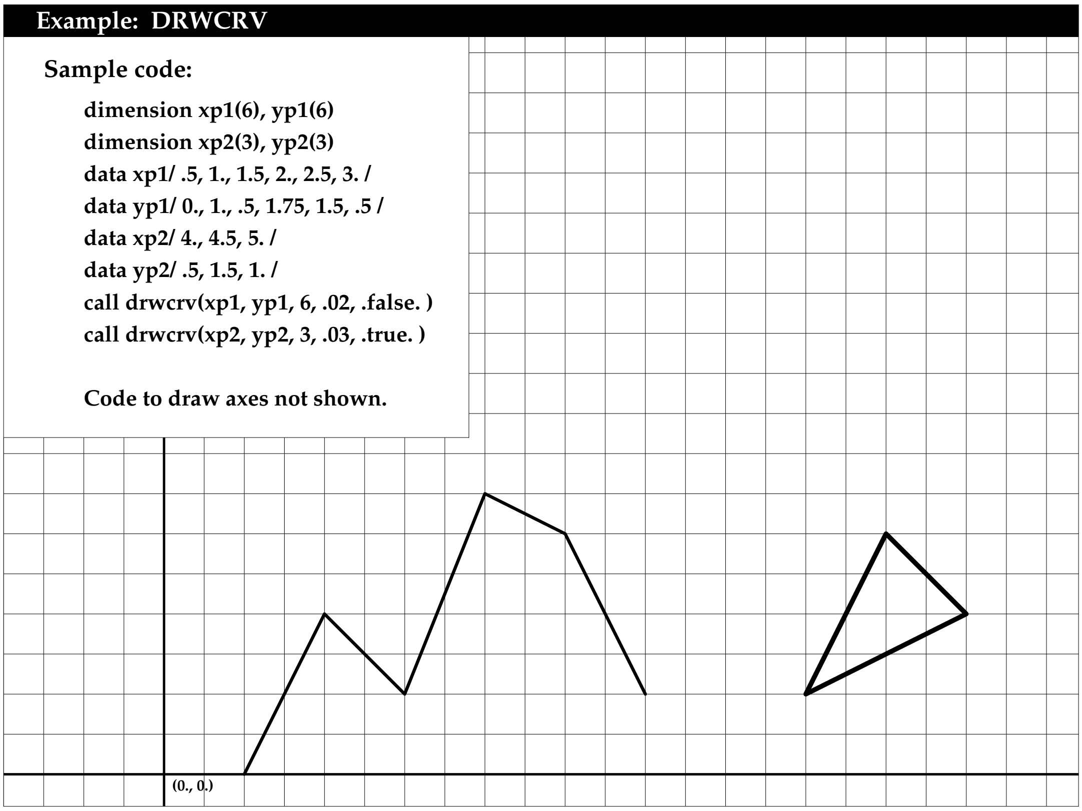
XARR	Array containing the x-coordinates of the points to be connected.
YARR	Array containing the y-coordinates of the points to be connected.
NPTS	Number of points in the curve.
THK	Thickness of the curve. If THK = 0., the current linewidth is used.
CLOSER	Logical variable. If CLOSER = .TRUE., the last point in the data arrays is connected to the first point, i.e. the curve is closed; otherwise, the curve is open.

Example: DRWCRV

Sample code:

```
dimension xp1(6), yp1(6)
dimension xp2(3), yp2(3)
data xp1/ .5, 1., 1.5, 2., 2.5, 3. /
data yp1/ 0., 1., .5, 1.75, 1.5, .5 /
data xp2/ 4., 4.5, 5. /
data yp2/ .5, 1.5, 1. /
call drwcrv(xp1, yp1, 6, .02, .false. )
call drwcrv(xp2, yp2, 3, .03, .true. )
```

Code to draw axes not shown.



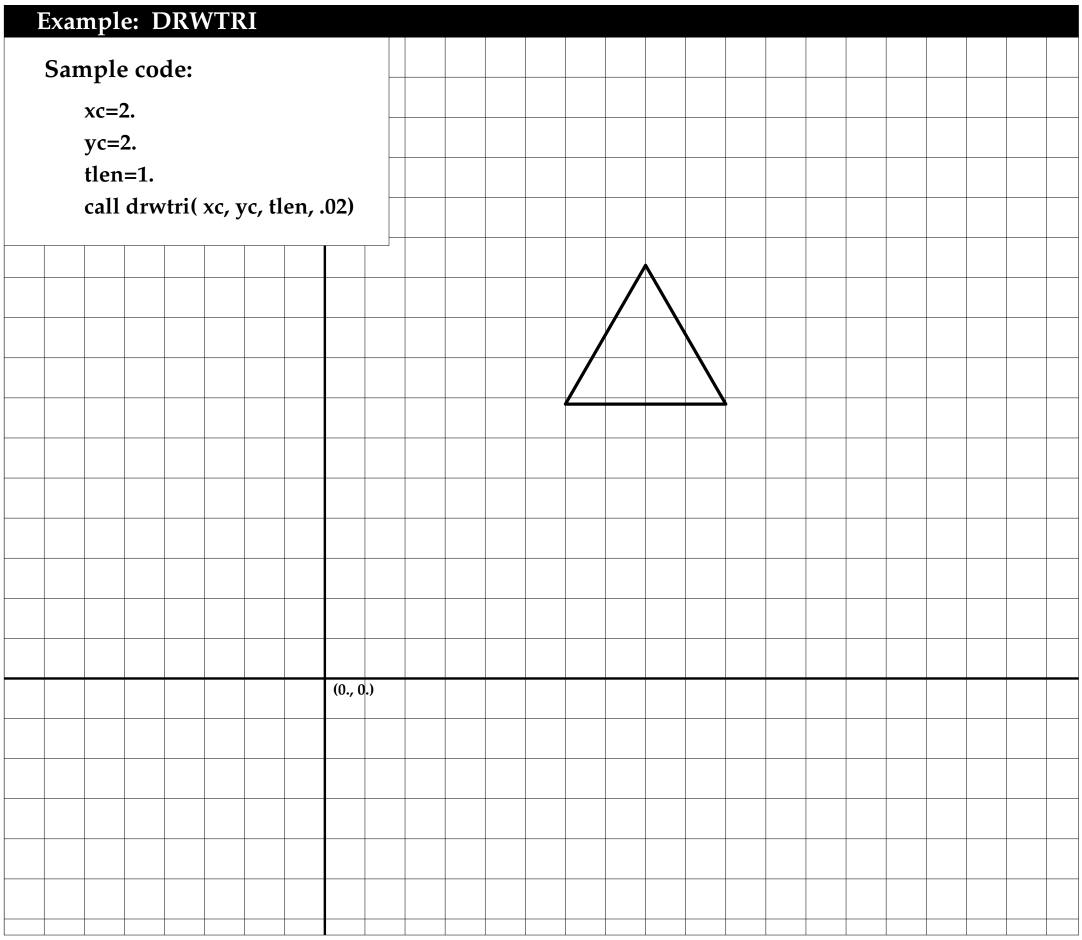
DRWTRI

PURPOSE **DRWTRI** draws an equilateral triangle.

SYNTAX CALL DRWTRI (XC, YC, SIDE, THK)

<u>*ARGUMENTS*</u>

XC, YC	X, Y coordinates of triangle **center**.
SIDE	Length of triangle side.
THK	Thickness of line used to draw the triangle. If THK = 0., the current line width is used.



DSHCRV

PURPOSE **DSHCRV** connects points in two data arrays using a line with a specified dash pattern.

SYNTAX CALL DSHCRV (XARR, YARR, NPTS, IDSHPN, THK)

ARGUMENTS

XARR Array containing the x-coordinates of the points to be connected.

YARR Array containing the y-coordinates of the points to be connected.

NPTS Number of points in the curve.

IDSHPN Dashed pattern of the line used to connect the data points. IDSHPN is specified as the number of times the dashed pattern (solid line/blank space) is repeated per inch.

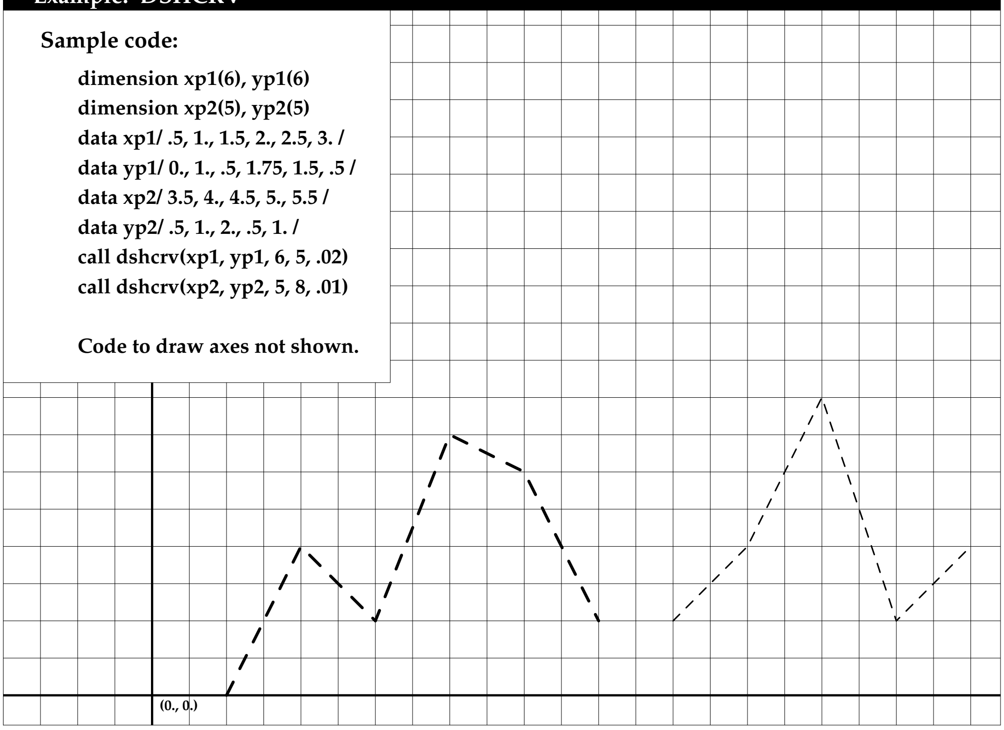
THK Thickness of the dashes in the curve. If THK = 0., the current line width is used.

Example: DSHCRV

Sample code:

```
dimension xp1(6), yp1(6)
dimension xp2(5), yp2(5)
data xp1/ .5, 1., 1.5, 2., 2.5, 3. /
data yp1/ 0., 1., .5, 1.75, 1.5, .5 /
data xp2/ 3.5, 4., 4.5, 5., 5.5 /
data yp2/ .5, 1., 2., .5, 1. /
call dshcrv(xp1, yp1, 6, 5, .02)
call dshcrv(xp2, yp2, 5, 8, .01)
```

Code to draw axes not shown.



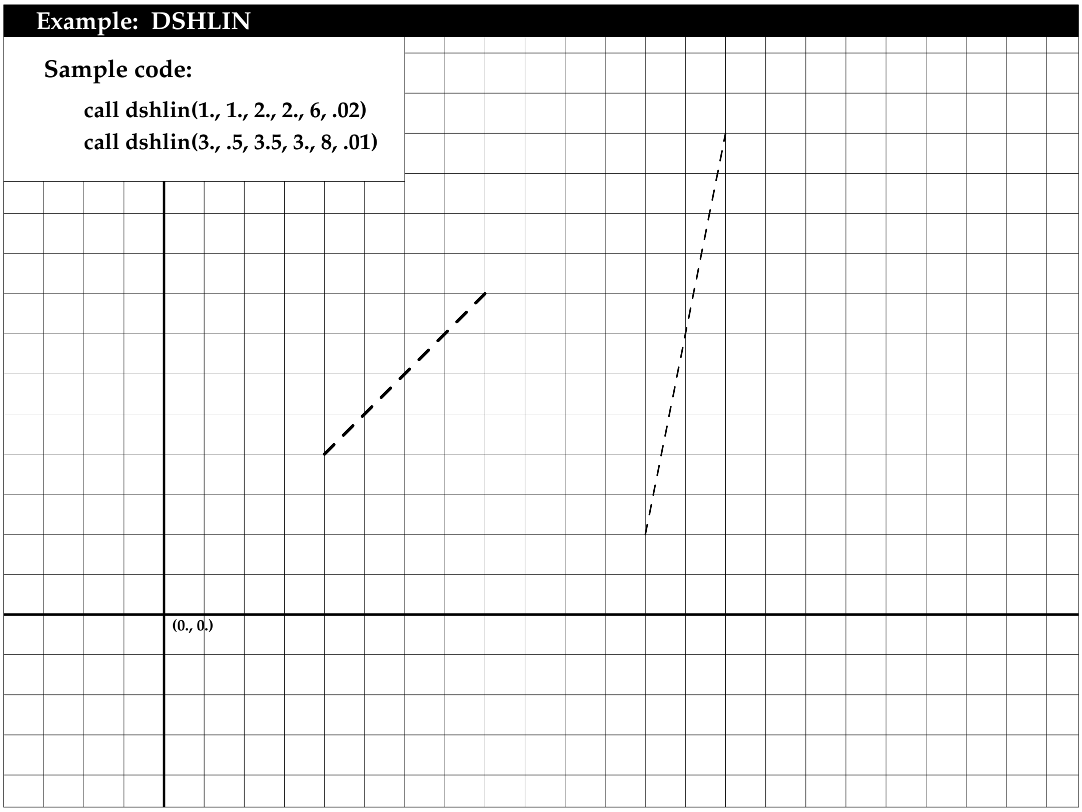
DSHLIN

PURPOSE **DSHLIN** connects two points using a line with a specified dash pattern.

SYNTAX CALL DSHLIN (X1, Y1, X2, Y2, IDSHPN, THK)

<u>*ARGUMENTS*</u>

X1,Y1	X,Y coordinates of first data point.
X2,Y2	X,Y coordinates of second data point.
IDSHPN	Dashed pattern of the line used to connect the two data points. IDSHPN is specified as the number of times the dashed pattern (solid line/blank space) is repeated per inch.
THK	Thickness of the dashes in the connecting line. If THK = 0., the current line width is used.



FACTOR

PURPOSE **FACTOR** enlarges or reduces the size of a plot by a specified ratio.

SYNTAX CALL FACTOR (FACT)

ARGUMENTS

FACT	The ratio of the desired plot size to the actual size. FACT must be a positive floating point number.

For example, to enlarge a plot to twice its normal size:

```
CALL FACTOR (2.0)
```

FAROHED

PURPOSE **FAROHED** draws a "fancy" arrowhead at a specified location.

SYNTAX CALL FAROHED (XPP, YPP, DIR, AROLNP, SPRANG, LOCXY, FILL)

ARGUMENTS

XPP,YPP X,Y coordinates of the tip of the arrowhead.

DIR Direction of arrowhead, measured east from north.

AROLNP Length of arrowhead sides.

SPRANG Half the angular spread of arrowhead.

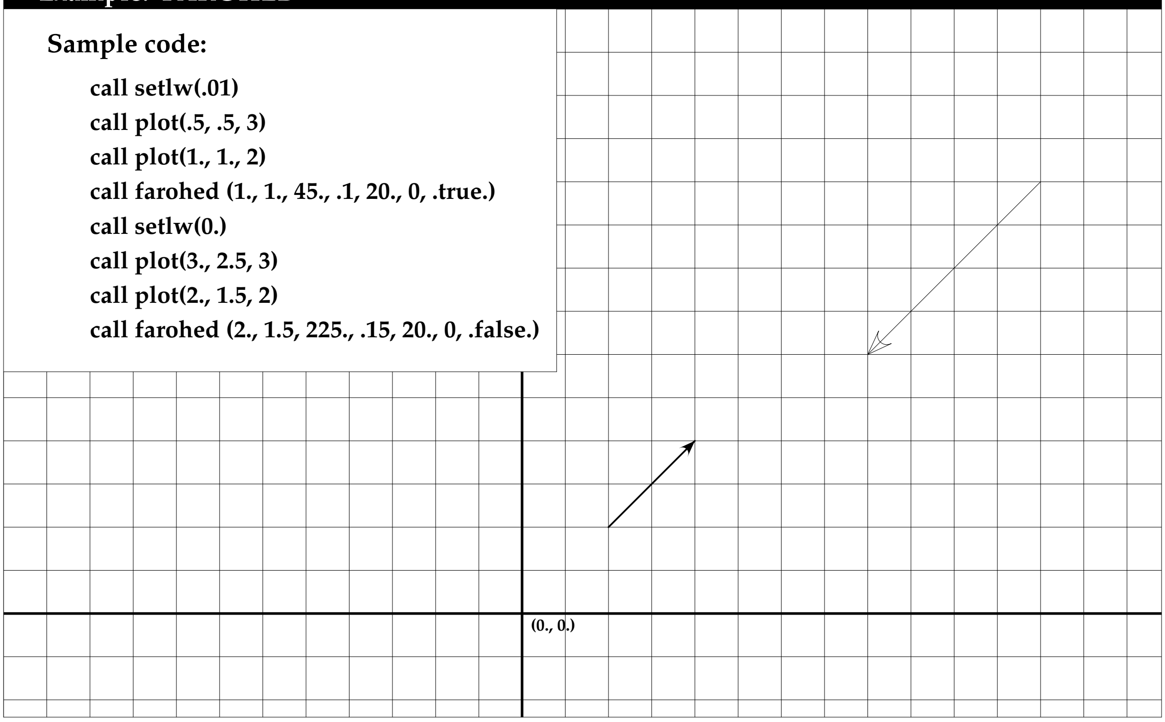
LOCXY Location of arrowhead point
- LOCXY=1 XPP,YPP at arrowhead point (most commonly used)
- LOCXY=2 XPP, YPP at center of arrowhead.
- LOCXY=3 XPP,YPP at tail of arrowhead.

FILL Logical variable. If FILL= .TRUE., the arrowhead is filled; otherwise it is outlined.

Example: FAROHED

Sample code:

```
call setlw(.01)
call plot(.5, .5, 3)
call plot(1., 1., 2)
call farohed (1., 1., 45., .1, 20., 0, .true.)
call setlw(0.)
call plot(3., 2.5, 3)
call plot(2., 1.5, 2)
call farohed (2., 1.5, 225., .15, 20., 0, .false.)
```



FILLBOX

PURPOSE **FILLBOX** fills a shape with a specified gray level.

SYNTAX CALL FILLBOX (XARR, YARR, NPTS, GRY)

ARGUMENTS

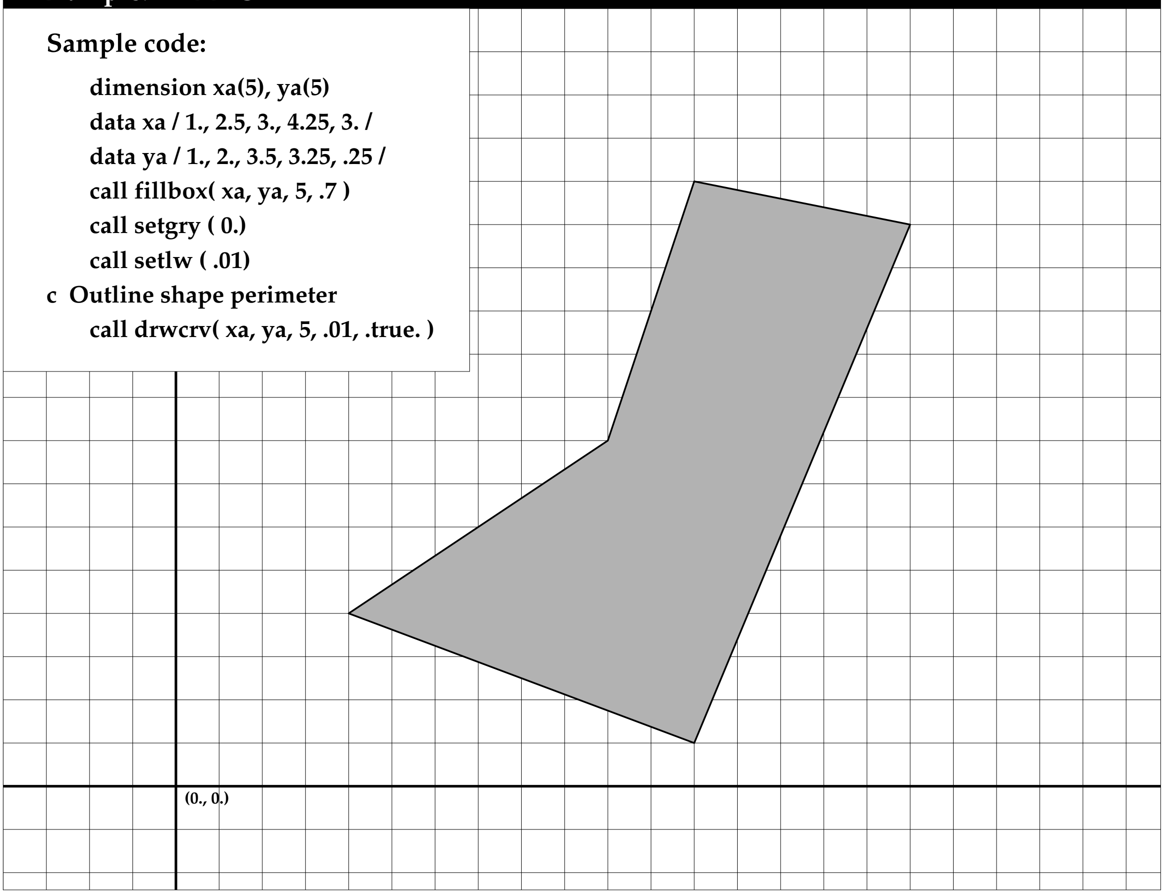
XARR	Array containing the x-coordinates of the points forming the perimeter of the shape to fill.
YARR	Array containing the y-coordinates of the points forming the perimeter of the shape to fill.
NPTS	Number of points comprising the shape perimeter.
GRY	Gray level of filled shape. GRY must lie between 0. (black) and 1. (white), inclusive.

Note: The current gray level remains at value GRY after exiting FILLBOX.

Example: FILLBOX

Sample code:

```
      dimension xa(5), ya(5)
      data xa / 1., 2.5, 3., 4.25, 3. /
      data ya / 1., 2., 3.5, 3.25, .25 /
      call fillbox( xa, ya, 5, .7 )
      call setgry ( 0.)
      call setlw ( .01)
c  Outline shape perimeter
      call drwcrv( xa, ya, 5, .01, .true. )
```



FILRGN

PURPOSE **FILRGN** is the same as FILLBOX, except it restores the current gray level after filling.

SYNTAX CALL FILRGN (XARR, YARR, NPTS, GRY)

ARGUMENTS

XARR Array containing the X coordinates of the points forming the perimeter of the shape to fill.

YARR Array containing the Y coordinates of the points forming the perimeter of the shape to fill.

NPTS Number of points comprising the shape perimeter.

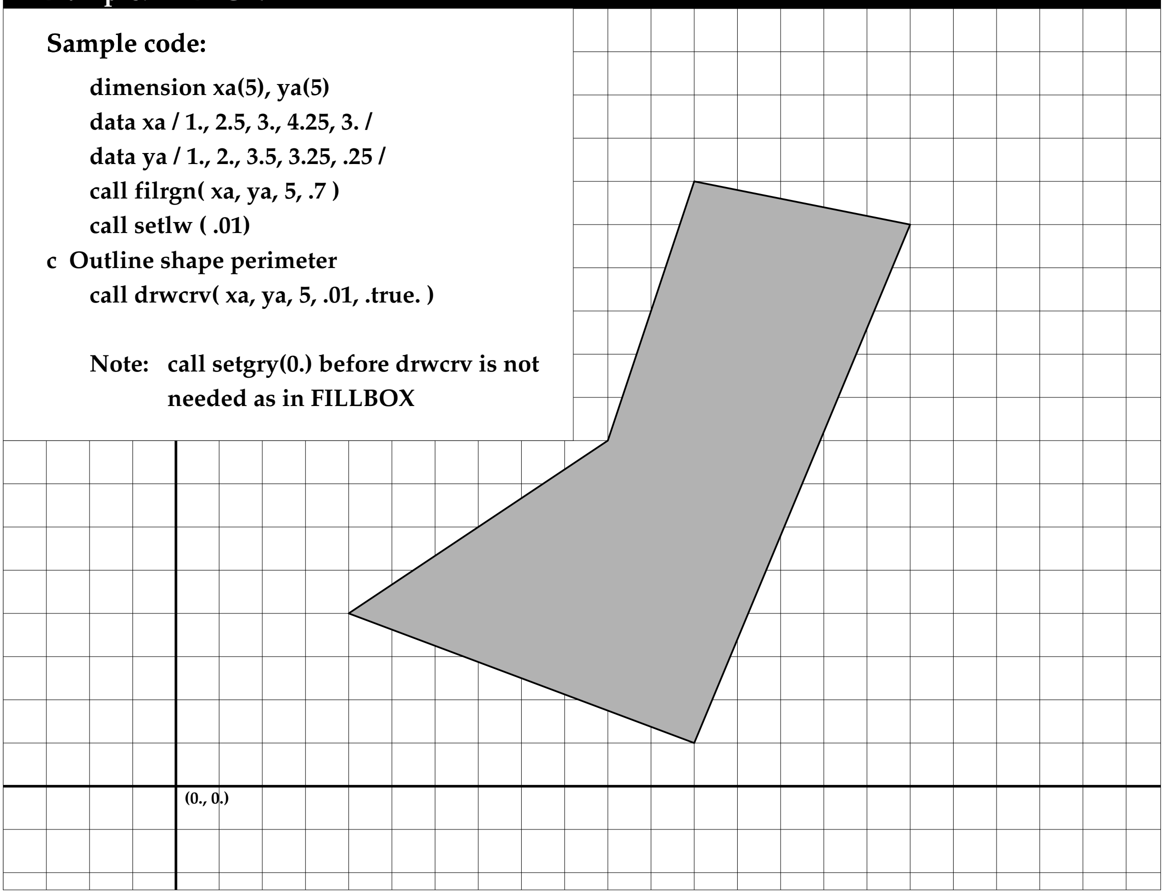
GRY Gray level of filled shape. GRY must lie between 0.(black) and 1. (white), inclusive.

Example: FILRGN

Sample code:

```
      dimension xa(5), ya(5)
      data xa / 1., 2.5, 3., 4.25, 3. /
      data ya / 1., 2., 3.5, 3.25, .25 /
      call filrgn( xa, ya, 5, .7 )
      call setlw ( .01)
c  Outline shape perimeter
      call drwcrv( xa, ya, 5, .01, .true. )

      Note:  call setgry(0.) before drwcrv is not
             needed as in FILLBOX
```



FILRGNC

PURPOSE **FILRGNC** fills a region with the current gray level or color.

SYNTAX CALL FILRGNC (XARR, YARR, NPTS)

ARGUMENTS

XARR Array containing the X coordinates of the points forming the perimeter of the shape to fill.

YARR Array containing the Y coordinates of the points forming the perimeter of the shape to fill.

NPTS Number of points comprising the shape perimeter.

GREST

PURPOSE **GREST** calls the PostScript operator *grestore*. It is useful after establishing and using a clipping region when it is no longer needed.

SYNTAX CALL GREST

ARGUMENTS

NONE

GREST is used to restore the graphics state of a PostScript page. It is called after a clipping region has been established and is no longer needed. GREST is used in conjunction with GSAV.

GRKSYM

PURPOSE **GRKSYM** plots a Greek symbol.

SYNTAX CALL GRKSYM (XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the Greek symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of Greek symbol to be plotted.

ICH Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below).

ANG Angle, measured counterclockwise from the X-axis, at which the character is to be plotted.

NCHAR Number of characters to be plotted (should be set to 1).

MJUS Controls the justification of the character to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the plotted character.

If MJUS = 1, (XPP,YPP) denotes the position of the center of the plotted character.

If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the plotted character.

PostScript Greek Symbols

Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	ϒ
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Φ	Χ	Ψ	Ω	α	β	γ	δ	ε	ζ	η	ϑ	ι	κ	λ	μ	ν	ξ	ο	π
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
ρ	σ	τ	υ	φ	χ	ψ	ω	θ	ϕ										
41	42	43	44	45	46	47	48	49	50										

GSAV

PURPOSE **GSAV** calls the PostScript operator *gsave*. It is useful before establishing and using a clipping region.

SYNTAX CALL GSAV

ARGUMENTS

NONE

GSAV is used to save the graphics state of the PostScript page prior to defining a clipping region. After the clipping region is used, GREST is called to restore the original graphics state.

HILITEC

PURPOSE **HILITEC** prints text surrounded by a rectangular box (color).

SYNTAX CALL HILITEC (XPP, YPP, HEIGHT, CSTR, ANG, EDG, JUSX, JUSY, FRED, FGREEN, FBLUE, BRED, BGREEN, BBLUE)

ARGUMENTS

XPP,YPP X,Y coordinates of the box surrounding the text.

HEIGHT Height of text to be plotted.

CSTR Character string text to be plotted.

ANG Angle, measured counterclockwise from the X-axis, at which the text is to be plotted.

EDG Fraction of the text height to use as an edge border.

JUSX Justification in the x-direction.
JUSX=0 is left justification.
JUSX=1 is centered.
JUSX=2 is right justification.

JUSY Similar to JUSX, but in the y-direction.
JUSY=0 is bottom justification.
JUSY=1 is centered.
JUSY=2 is top justification.

FRED, FGREEN, FBLUE Foreground text RGB values. Must lie between 0. and 1.0, inclusive.

BRED, BGREEN, BBLUE Background box RGB values. Must lie between 0. and 1.0, inclusive.

See code example for HILITEG.

HILITEG

PURPOSE **HILITEG** prints text surrounded by a rectangular box (grayscale).

SYNTAX CALL HILITEG (XPP, YPP, HEIGHT, CSTR, ANG, EDG, JUSX, JUSY, FGRY, BGRY)

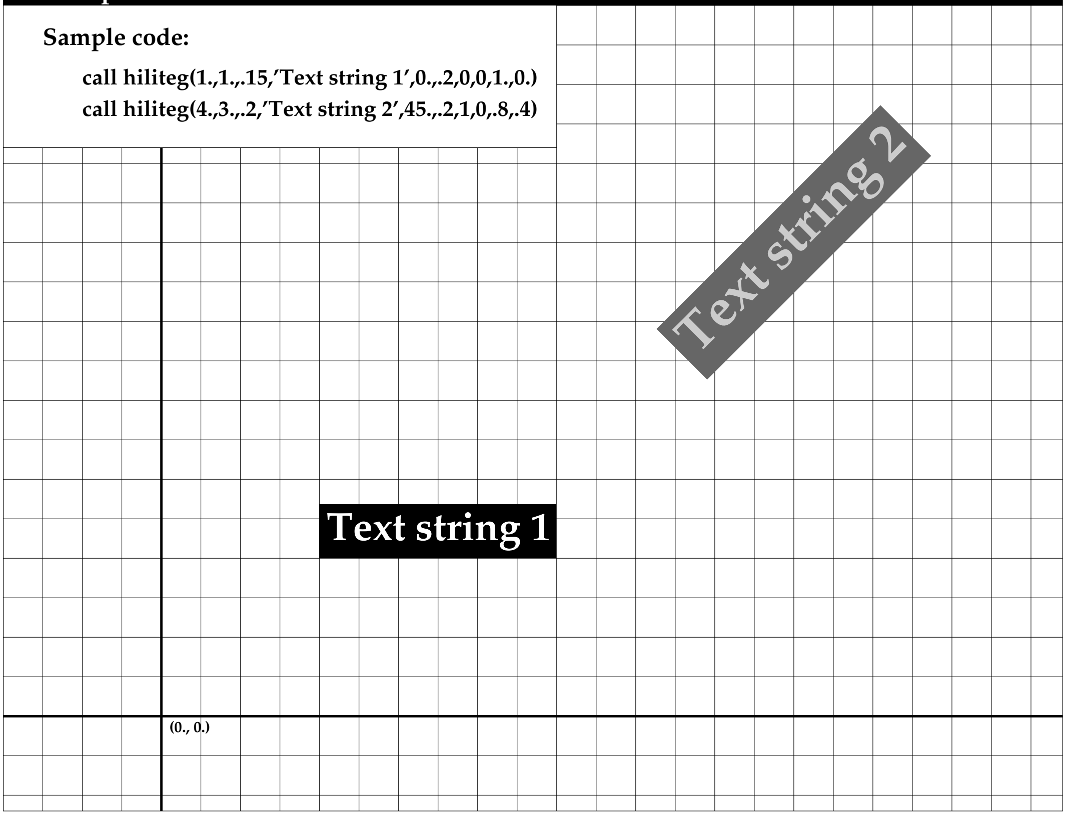
<u>*ARGUMENTS*</u>

XPP,YPP	X,Y coordinates of the box surrounding the text.
HEIGHT	Height of text to be plotted.
CSTR	Character string text to be plotted.
ANG	Angle, measured counterclockwise from the X-axis, at which the text is to be plotted.
EDG	Fraction of the text height to use as an edge border.
JUSX	Justification in the x-direction. JUSX=0 is left justification. JUSX=1 is centered. JUSX=2 is right justification.
JUSY	Similar to JUSX, but in the y-direction. JUSY=0 is bottom justification. JUSY=1 is centered. JUSY=2 is top justification.
FGRY	Graylevel of the (foreground) text. FGRY must lie between 0. and 1.0, inclusive.
BGRY	Graylevel of the (background) surrounding box. BGRY must lie between 0. and 1.0, inclusive.

Example: HILITEG

Sample code:

```
call hiliteg(1.,1.,.15,'Text string 1',0.,.2,0,0,1.,0.)
call hiliteg(4.,3.,.2,'Text string 2',45.,.2,1,0,.8,.4)
```



INTEGRAL

PURPOSE **INTEGRAL** plots an integral with upper and lower limits.

SYNTAX CALL INTEGRAL (XPP, YPP, HEIGHT, ANG, LOWER, NL, LUPPER, NU)

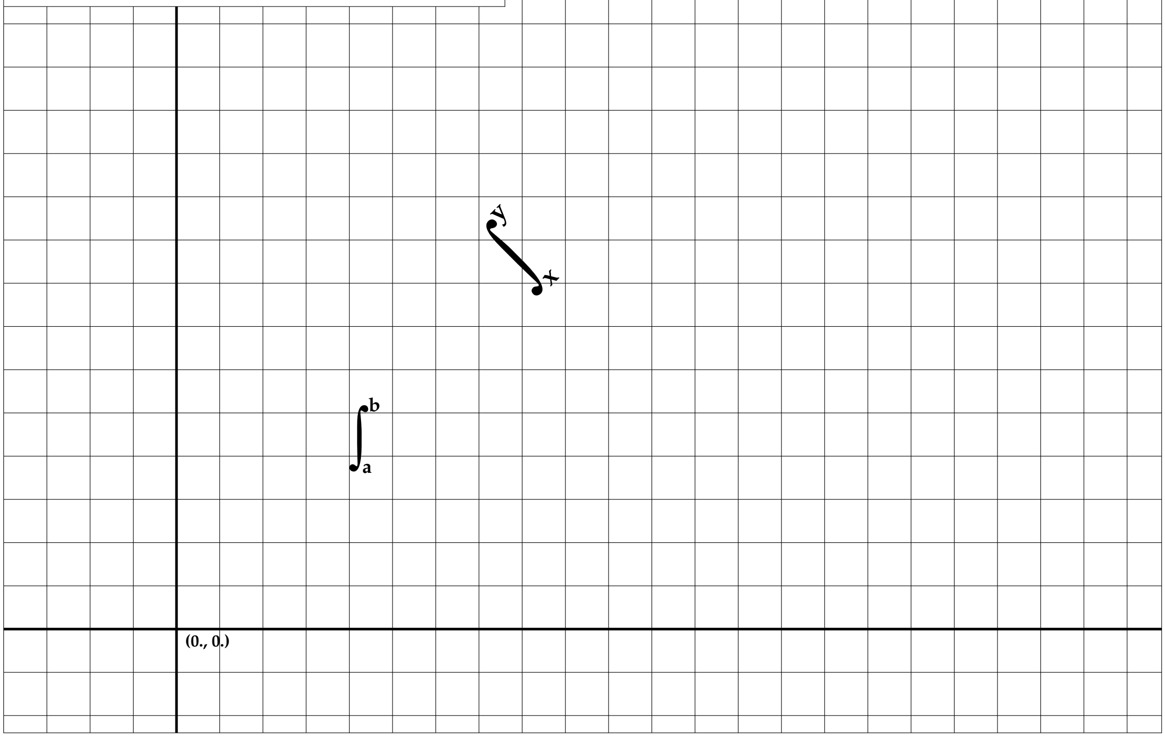
<u>*ARGUMENTS*</u>

XPP,YPP	X,Y coordinates of the lower left edge of the integral.
HEIGHT	Height of the integral.
ANG	Angle, measured counterclockwise from the X-axis, at which the integral is plotted.
LOWER	Character string of the lower limit (Hollerith).
NL	Number of characters in the lower limit text.
LUPPER	Character string of the upper limit (Hollerith).
NU	Number of characters in the upper limit text.

Example: INTEGRAL

Sample code:

```
call integral(1.,1.,.15,0.,1ha,1,1hb,1)
call integral(2.,2.,.2,45.,1hx,1,1hy,1)
```



KEKEXP

PURPOSE **KEKEXP** plots a floating point number in exponential format.

SYNTAX CALL KEKEXP (XPP, YPP, HEIGHT, FNUM, ANG, NDEC, MJUS)

ARGUMENTS

XPP,YPP — X,Y coordinates of the number to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation

HEIGHT — Height of number to be plotted.

FNUM — Floating point number to be plotted.

ANG — Angle, measured counterclockwise from the X-axis, at which the number is to be plotted.

NDEC — Controls the plotted precision of FNUM.

If NDEC > 0, it specifies the number of digits to the right of the decimal point, after rounding.
If NDEC = 0, only the number's integer portion and a decimal point are plotted, after rounding.
If NDEC = -1, only the number's integer portion is plotted, after rounding.

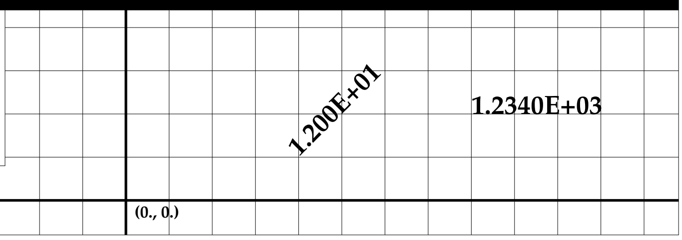
MJUS — Controls the justification of the number to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the plotted number.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the plotted number.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the plotted number.

Example: KEKEXP

Sample code:

```
call kekexp(1., .25, .1, 12., 45., 3, 0 )
call kekexp(2., .5, .1, 1234., 0., 4, 0 )
```



KEKFLT

PURPOSE **KEKFLT** is the same as KEKNUM.

SYNTAX CALL KEKFLT (XPP, YPP, HEIGHT, FNUM, ANG, NDEC, MJUS)

ARGUMENTS

Same as KEKNUM.

KEKNUM

PURPOSE **KEKNUM** plots a floating point number in floating point format.

SYNTAX CALL KEKNUM (XPP, YPP, HEIGHT, FNUM, ANG, NDEC, MJUS)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the number to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of number to be plotted.

FNUM Floating point number to be plotted.

ANG Angle, measured counterclockwise from the X-axis, at which the number is to be plotted.

NDEC Controls the plotted precision of FNUM.

If NDEC > 0, it specifies the number of digits to the right of the decimal point, after rounding.
If NDEC = 0, only the number's integer portion and a decimal point are plotted, after rounding.
If NDEC = -1, only the number's integer portion is plotted, after rounding.

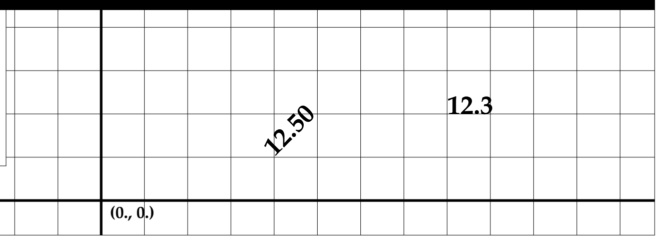
MJUS Controls the justification of the number to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the plotted number.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the plotted number.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the plotted number.

Example: KEKNUM

Sample code:

```
call keknum(1., .25, .1, 12.5, 45., 2, 0 )
call keknum(2., .5, .1, 12.34, 0., 1, 0 )
```



KEKSYM

PURPOSE **KEKSYM** plots a character string.

SYNTAX CALL KEKSYM (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS)

ARGUMENTS

XPP,YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

Special feature of KEKSYM:

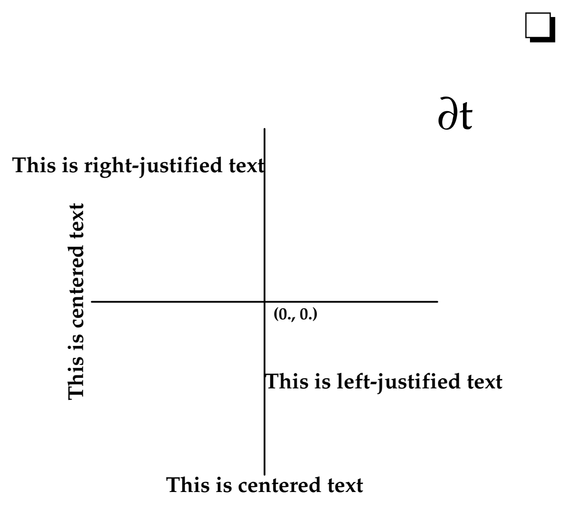
You can plot special characters, for example the characters in fonts Symbol and Zapf Dingbats, by setting NCHAR=-999 and IBCD equal to the octal code of the character you want to plot.

Example: KEKSYM

Sample code:

```
 call keksym(0., -1.1, .08,
1 21hThis is centered text, 0., 21, 1 )
 call keksym(-1.05, 0., .08,
1 21hThis is centered text, 90., 21, 1 )
 call keksym( 0.,-.5, .08,
1 27hThis is left-justified text, 0., 27, 0 )
 call keksym(0., .75, .08,
1 28hThis is right-justified text, 0., 28, 2 )

call setfnt(29)   !Sets font to Symbol
call keksym(1., 1., .15, 266, 0., -999, 0)
call setfnt(28)   !Sets font to Palatino-Roman
call keksym(999., 999., .15, 1ht, 0., 1, 0)
call setfnt(35)   !Sets font to Zapf Dingbats
call keksym(1.5, 1.5, .15, 157, 0., -999, 0)
```



KEKSYMC

PURPOSE **KEKSYMC** plots a character string. Same as **KEKSYM**, but text is input as a character string rather than Hollerith.

SYNTAX CALL KEKSYMC (XPP, YPP, HEIGHT, CSTR, ANG, NCHAR, MJUS)

ARGUMENTS

XPP,YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of character string to be plotted.

CSTR Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

Special feature of KEKSYMC:

You can plot special characters, for example the characters in fonts Symbol and Zapf Dingbats, by setting NCHAR=-999 and CSTR equal to the octal code of the character you want to plot. CSTR must be in single quotes.

KEKSYMO

PURPOSE **KEKSYMO** plots a character string; however, the characters are outlined rather than filled (solid).

SYNTAX CALL KEKSYMO (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS)

ARGUMENTS

XPP,YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

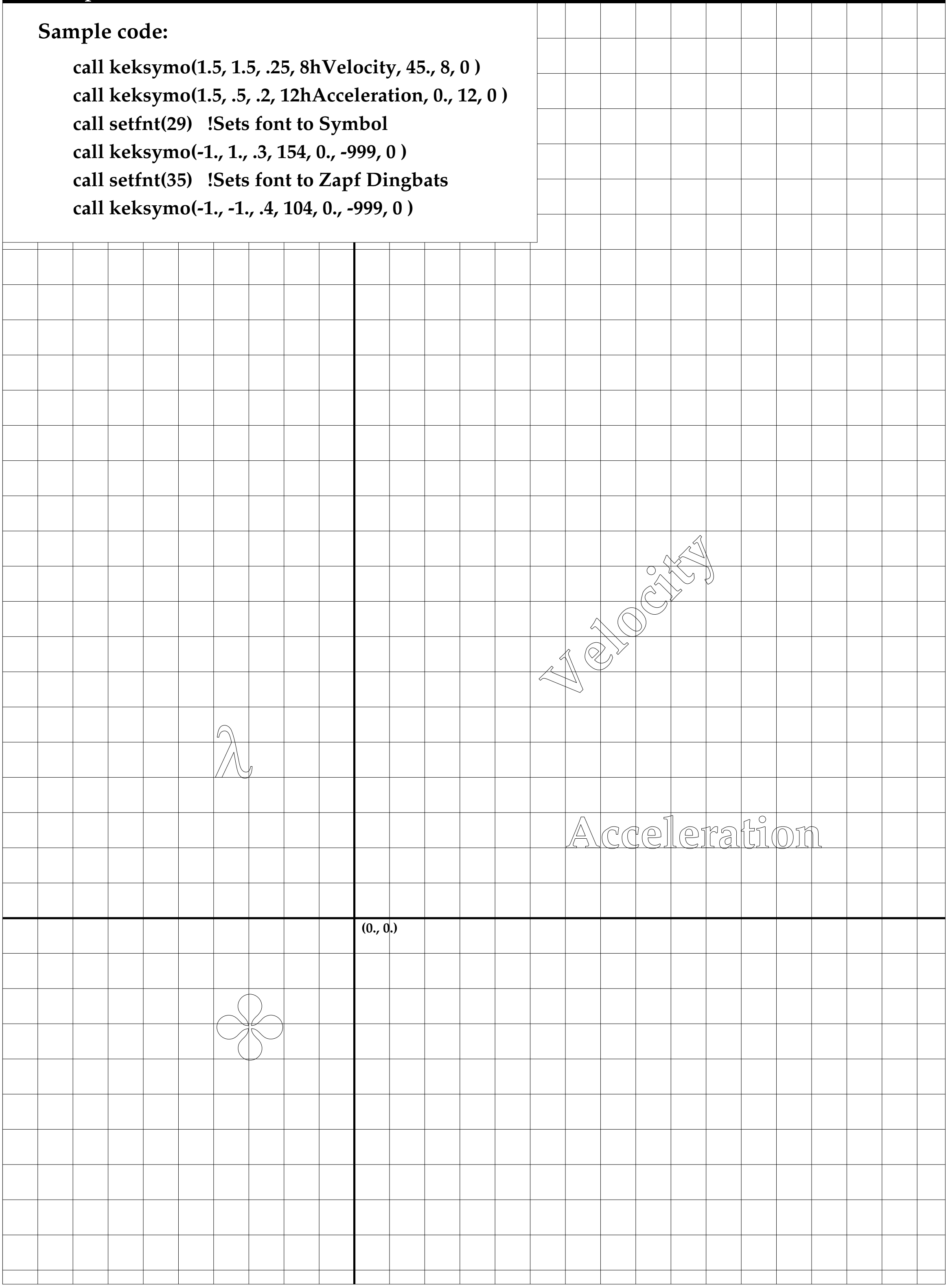
Special feature of KEKSYMO:

You can plot special characters, for example the characters in fonts Symbol and Zapf Dingbats, by setting NCHAR=-999 and IBCD equal to the octal code of the character you want to plot.

Example: KEKSYMO

Sample code:

```
call keksymo(1.5, 1.5, .25, 8hVelocity, 45., 8, 0 )
call keksymo(1.5, .5, .2, 12hAcceleration, 0., 12, 0 )
call setfnt(29)   !Sets font to Symbol
call keksymo(-1., 1., .3, 154, 0., -999, 0 )
call setfnt(35)   !Sets font to Zapf Dingbats
call keksymo(-1., -1., .4, 104, 0., -999, 0 )
```



LENSTR

PURPOSE	**LENSTR** is a function whose returned value is the number of characters in the character string argument after trailing blanks have been removed.
SYNTAX	LS = LENSTR(STRING, NDIM)

<u>*ARGUMENTS*</u>

STRING	Character string variable.
NDIM	Maximum length of STRING as stated in calling program.
LS	Returned length of the character string STRING after trailing blanks have been removed.

NEWDEV

PURPOSE **NEWDEV** specifies a non-default filename which will contain the application program's output PostScript plot file.

SYNTAX CALL NEWDEV (FLNAME, NCHAR)

ARGUMENTS

FLNAME Character string or FORTRAN character variable which contains the name of the output Postscript file.

NCHAR Number of characters in the filename.

For example,

```
CALL NEWDEV ('MYPLOT.DAT', 10)
```

NEWDEV is called only once per plotting session and must be called prior to PSINIT. If NEWDEV is not called, the name of the output PostScript file is **psplot.ps**.

NUMBER

PURPOSE **NUMBER** is the same as KEKNUM with left justification.

SYNTAX CALL NUMBER (XPP, YPP, HEIGHT, FNUM, ANG, NDEC)

ARGUMENTS

Same as KEKNUM.

ONEHLF

PURPOSE **ONEHLF** plots the symbol ½ .

SYNTAX CALL ONEHLF (XPP, YPP, HEIGHT, ANG, MJUS)

ARGUMENTS

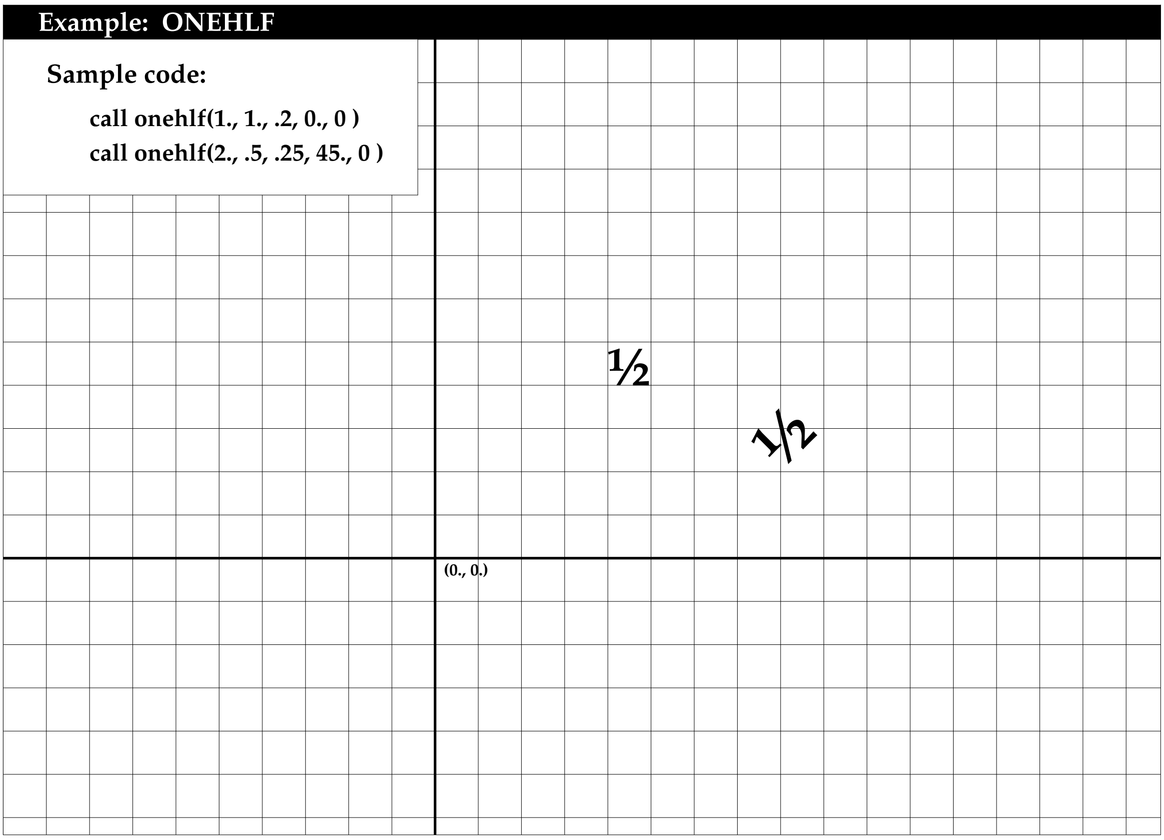
XPP,YPP X,Y coordinates of the symbol to be plotted.

HEIGHT Height of symbol to be plotted.

ANG Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.



OVERBAR

PURPOSE **OVERBAR** plots a character string with an overbar.

SYNTAX CALL OVERBAR (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of number to be plotted.

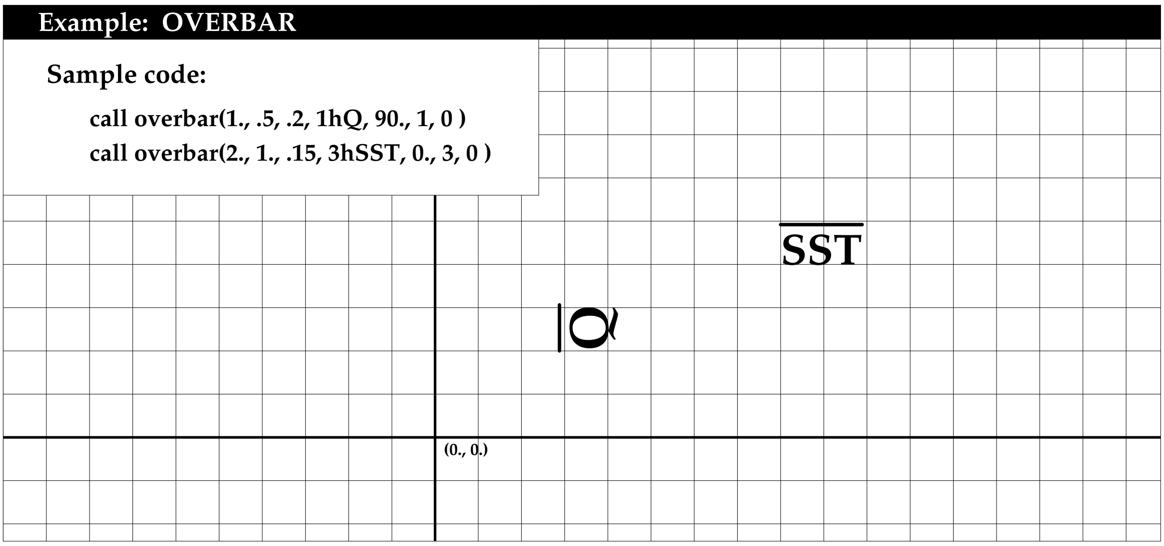
IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.



OVERSBSP

PURPOSE **OVERSBSP** plots a subscripted and superscripted character string with an overbar. The overbar extends to cover both the subscripts and superscripts.

SYNTAX CALL OVERSBSP (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS,ISUB,NSUB, ISUP, NSUP)

<u>ARGUMENTS</u>

XPP,YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

ISUB Hollerith character(s) comprising the subscript.

NSUB Number of characters in the subscript string.

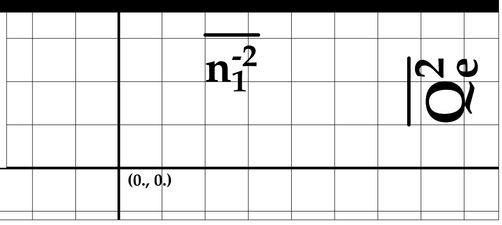
ISUP Hollerith character(s) comprising the superscript.

NSUP Number of characters in the superscript string.

Example: OVERSBSP

Sample code:

```
call oversbsp(2., .25, .2, 1hQ, 90., 1, 0, 1he, 1, 1h2, 1 )
call oversbsp(.5, .5, .15, 1hn, 0., 1, 0, 1h1, 1, 2h-2, 2 )
```



OVERSBSPG

PURPOSE **OVERSBSPG** plots a subscripted and superscripted Greek symbol with an overbar. The overbar extends to cover both the subscripts and superscripts.

SYNTAX CALL OVERSBSPG (XPP, YPP, HEIGHT, ICH, ANG, CHAR, MJUS, ISUB, NSUB, ISUP, NSUP)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of symbol to be plotted.

ICH Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below Subroutine GRKSYM).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot (should be set to 1).

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

ISUB Hollerith character(s) comprising the subscript.

NSUB Number of characters in the subscript string.

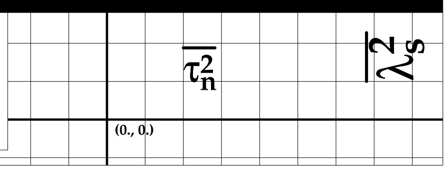
ISUP Hollerith character(s) comprising the superscript.

NSUP Number of characters in the superscript string.

Example: OVERSBSPG

Sample code:

```
call oversbspg(2., .25, .2, 35, 90., 1, 0, 1hs, 1, 1h2, 1 )
call oversbspg(.5, .25, .15, 43, 0., 1, 0, 1hn, 1, 1h2, 1 )
```



OVERSUB

PURPOSE **OVERSUB** plots a subscripted character string with an overbar. The overbar extends to cover the subscripts.

SYNTAX CALL OVERSUB (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR, MJUS, ISUB, NSUB)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

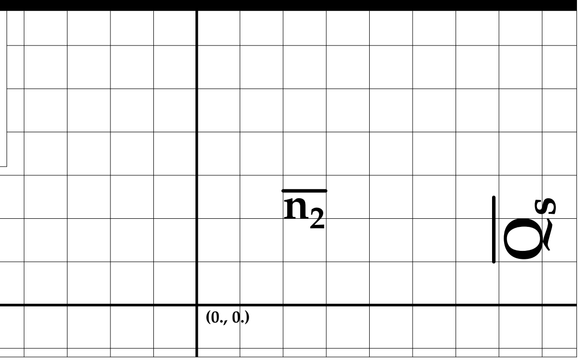
ISUB Hollerith character(s) comprising the subscript.

NSUB Number of characters in the subscript string.

Example: OVERSUB

Sample code:

```
call oversub(2., .25, .2, 1hQ, 90., 1, 0, 1hs, 1 )
call oversub(.5, .5, .15, 1hn, 0., 1, 0, 1h2, 1 )
```



OVERSUBG

PURPOSE **OVERSUBG** plots a subscripted Greek symbol with an overbar. The overbar extends to cover the subscript.

SYNTAX CALL OVERSUBG(XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS,ISUB,NSUB)

ARGUMENTS

XPP,YPP — X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT — Height of the Greek symbol to be plotted.

ICH — Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below Subroutine GRKSYM).

ANG — Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

NCHAR — Number of characters in the string to plot (should be set to 1).

MJUS — Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the symbol.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the symbol.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the symbol.

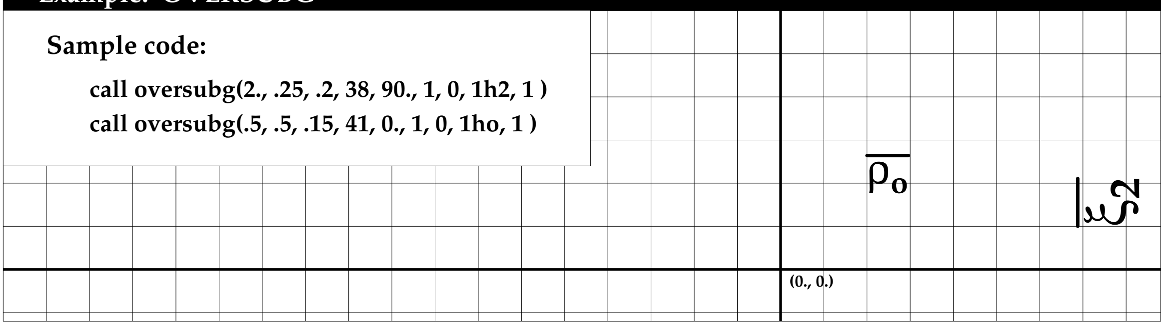
ISUB — Hollerith character(s) comprising the subscript.

NSUB — Number of characters in the subscript string.

Example: OVERSUBG

Sample code:

```
call oversubg(2., .25, .2, 38, 90., 1, 0, 1h2, 1 )
call oversubg(.5, .5, .15, 41, 0., 1, 0, 1ho, 1 )
```



OVERSUP

PURPOSE **OVERSUP** plots a superscripted character string with an overbar. The overbar extends to cover the subscript.

SYNTAX CALL OVERSUP (XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS,ISUP, NSUP)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the character string to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string to plot.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character string.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character string.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character string.

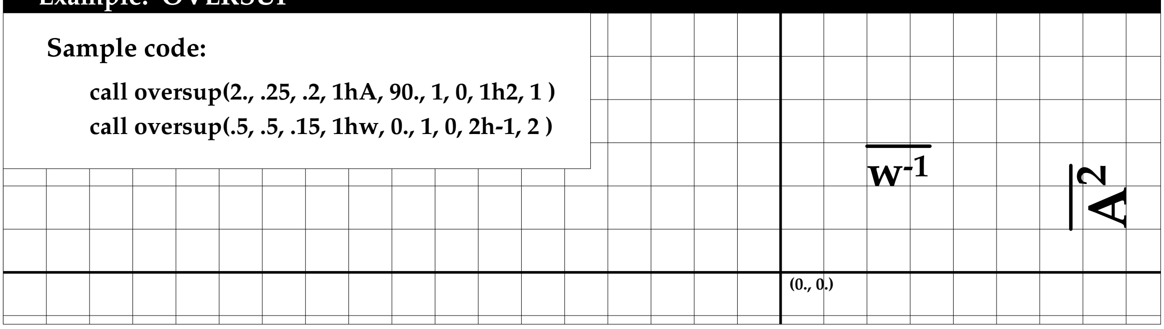
ISUP Hollerith character(s) comprising the superscript.

NSUP Number of characters in the superscript string.

Example: OVERSUP

Sample code:

```
call oversup(2., .25, .2, 1hA, 90., 1, 0, 1h2, 1 )
call oversup(.5, .5, .15, 1hw, 0., 1, 0, 2h-1, 2 )
```



OVERSUPG

PURPOSE **OVERSUPG** plots a subscripted Greek character with an overbar. The overbar extends to cover the superscript.

SYNTAX CALL OVERSUPG(XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS, ISUP, NSUP)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the Greek symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of the Greek symbol to be plotted.

ICH Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below Subroutine GRKSYM).

ANG Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

NCHAR Number of characters in the string to plot (should be set to 1).

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the symbol.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the symbol.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the symbol.

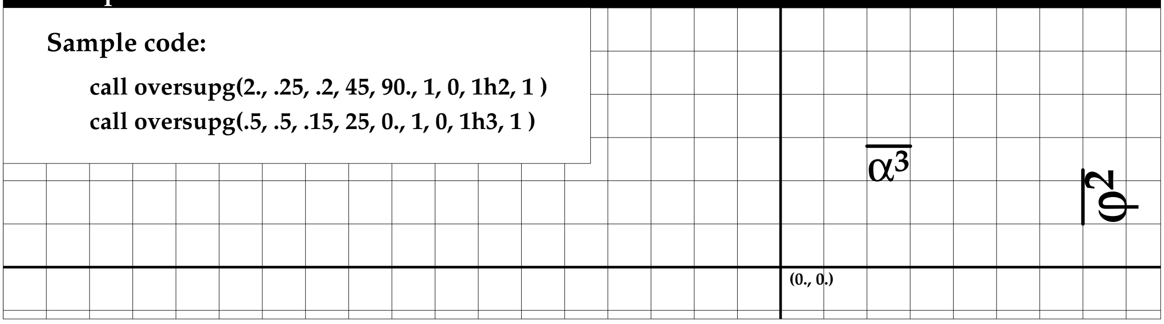
ISUP Hollerith character(s) comprising the superscript.

NSUP Number of characters in the superscript string.

Example: OVERSUPG

Sample code:

```
call oversupg(2., .25, .2, 45, 90., 1, 0, 1h2, 1 )
call oversupg(.5, .5, .15, 25, 0., 1, 0, 1h3, 1 )
```



OVRGRK

PURPOSE **OVRGRK** plots a Greek symbol with an overbar.

SYNTAX CALL OVRGRK (XPP, YPP, HEIGHT, ICH, ANG, NCHAR, MJUS)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the Greek symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of symbol to be plotted.

ICH Integer value between 1 and 50 corresponding to the desired Greek symbol (see table below Subroutine GRKSYM).

ANG Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

NCHAR Number of characters in the string to plot (should be set to 1).

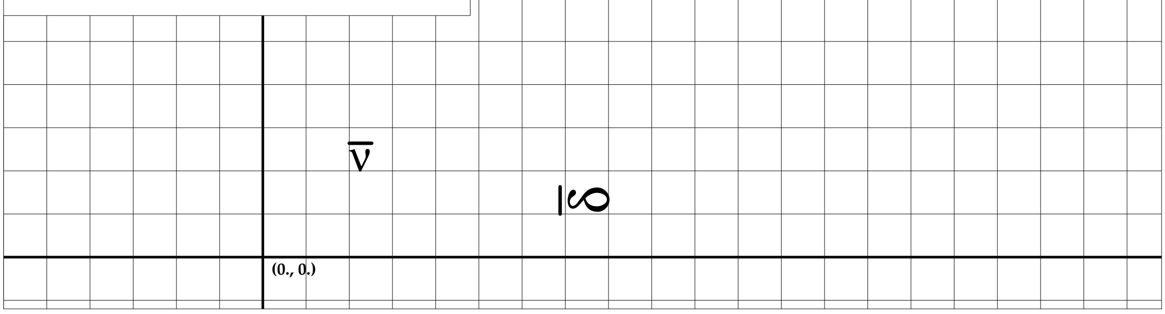
MJUS Controls the justification of the symbol to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the symbol.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the symbol.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the symbol.

Example: OVRGRK

Sample code:

```
call ovrgrk(2., .25, .2, 28, 90., 1, 0 )
call ovrgrk(.5, .5, .15, 37, 0., 1, 0 )
```



PLOT

PURPOSE **PLOT** is the most fundamental user-level plotting command. It gives you direct control of pen movement (to any X,Y coordinate position) and pen status (up or down). Additionally, it allows you to re-define the current plotting origin.

SYNTAX CALL PLOT (XPP, YPP, IPEN)

<u>*ARGUMENTS*</u>

XPP,YPP X,Y coordinates of the position to which the pen is to be moved. An origin may be established anywhere on or off the plotting surface, as explained below for negative IPEN values.

IPEN A signed integer which controls pen status (up/down) and origin definition.

If IPEN = 2, the pen is down during movement, thus drawing a visible line.

If IPEN = 3, the pen is up during movement, thus changing the pen's current position only.

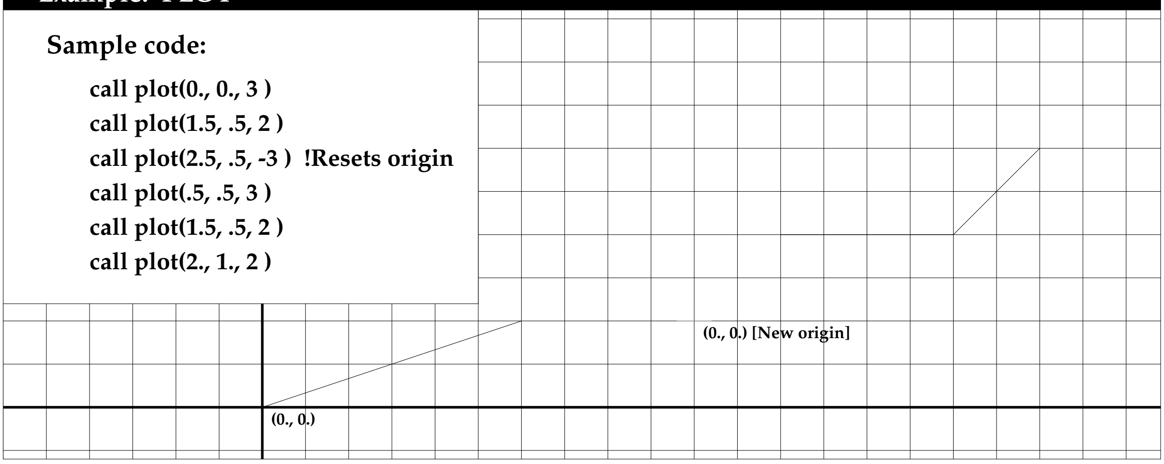
If IPEN = -2 or -3, a new origin is defined at the position (XPP,YPP) after the movement is completed as if IPEN were positive. The logical X,Y coordinates of the new pen position are set to (0,0), so that all subsequent pen movements use this position as a reference point.

If IPEN = 999, the call to PLOT closes the output file. Thus, a call to PLOT with IPEN = 999 may be used only once in a given plotting session, and if used, must be the last plotting command in the plotting session. Calling PLOT with IPEN = 999 is identical to calling PLOTND (which see).

Example: PLOT

Sample code:

```
call plot(0., 0., 3 )
call plot(1.5, .5, 2 )
call plot(2.5, .5, -3 )  !Resets origin
call plot(.5, .5, 3 )
call plot(1.5, .5, 2 )
call plot(2., 1., 2 )
```



PLOTND

PURPOSE PLOTND closes the output PostScript plot file to terminate the current plot defined by the user application program.

SYNTAX CALL PLOTND

ARGUMENTS

None

You ***must*** call PLOTND (or PLOT (0., 0., 999)) as the last plotting call in the application program.

PLSMIN

PURPOSE **PLSMIN** plots the symbol ±.

SYNTAX CALL PLSMIN (XPP, YPP, HEIGHT, ANG, MJUS)

ARGUMENTS

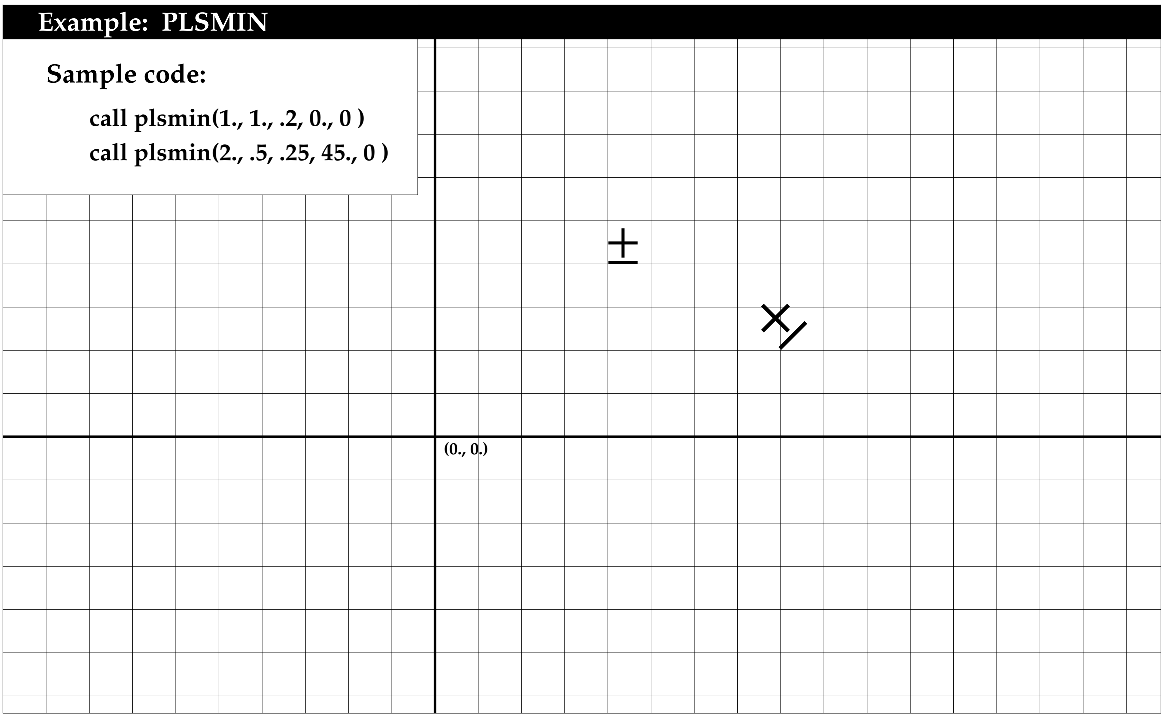
XPP,YPP X,Y coordinates of the symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of symbol to be plotted.

ANG Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

MJUS Controls the justification of the character string to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character.



PRIME

PURPOSE **PRIME** plots the symbol ′ .

SYNTAX CALL PRIME (XPP, YPP, HEIGHT, ANG, MJUS)

ARGUMENTS

XPP,YPP X,Y coordinates of the symbol to be plotted. Plotting may be continued from the end of a previously plotted character when used in conjunction with any of the subroutines which support continuation.

HEIGHT Height of symbol to be plotted.

ANG Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.

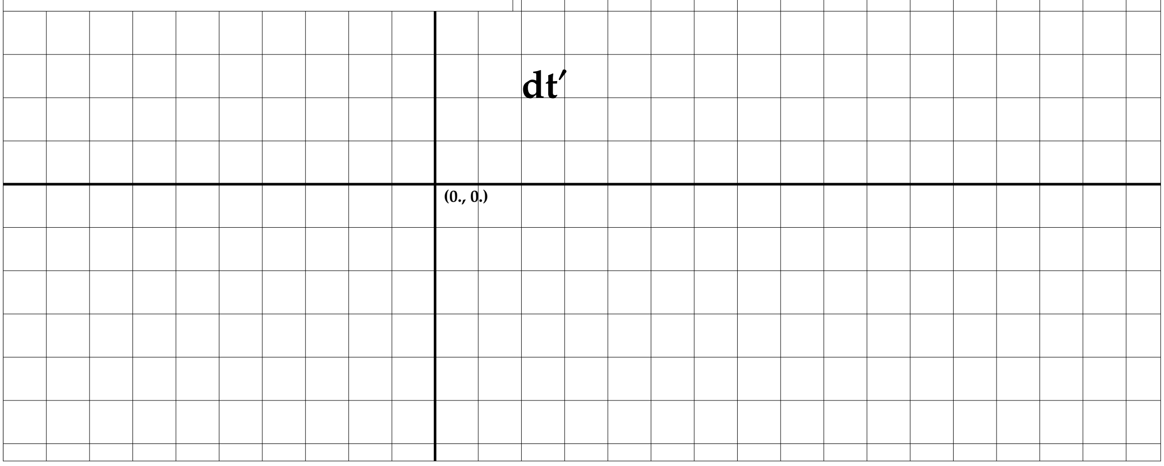
MJUS Controls the justification of the symbol to be plotted.

If MJUS = 0, (XPP,YPP) denotes the position of the lower left corner of the character.
If MJUS = 1, (XPP,YPP) denotes the position of the center of the character.
If MJUS = 2, (XPP,YPP) denotes the position of the lower right corner of the character.

Example: PRIME

Sample code:

```
call keksym( .5, .5, .14, 2hdt, 0., 2, 0 )
call prime( 999., 999., .14, 0., 0)
```



PSINIT

PURPOSE **PSINIT** is called to begin a plotting session. It must be called before any other plotting command, with the exception of NEWDEV.

SYNTAX CALL PSINIT (PRTRT)

ARGUMENTS

PRTRT Logical variable indicating the paper orientation. If PRTRT=.TRUE., the paper is oriented in portrait mode, i.e. long side of page is vertical. If PRTRT=.false., the paper is oriented in landscape mode, i.e. long side of page is horizontal.

RECT

PURPOSE **RECT** draws a rectangle (or square).

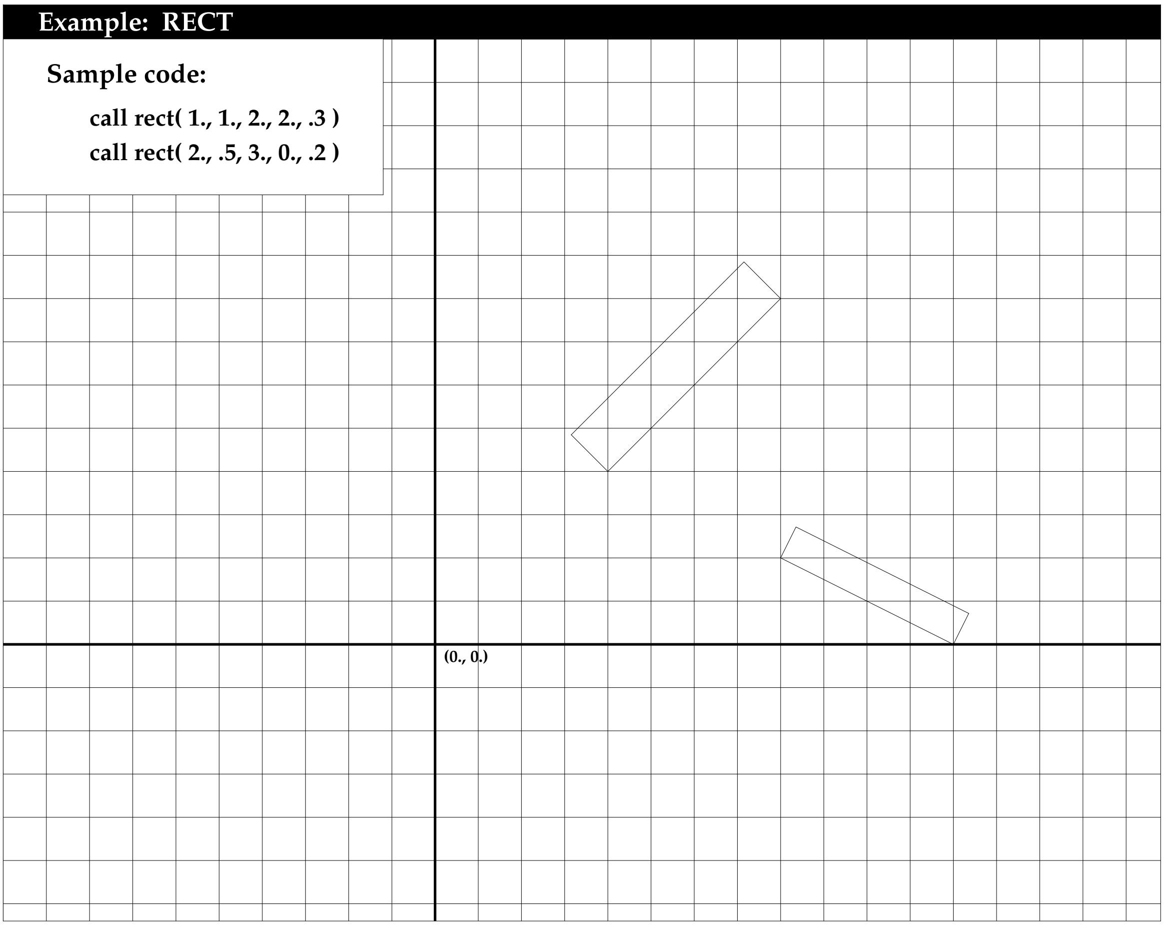
SYNTAX CALL RECT (XX1, YY1, XX2, YY2, HEIGHT)

ARGUMENTS

XX1,YY1 X,Y coordinates of lower left corner of rectangle.

XX2,YY2 X,Y coordinates of lower right corner of rectangle.

HEIGHT Height of rectangle.



RECTFILC

PURPOSE **RECTFILC** draws a rectangle (or square) and fills it with a specified red, green, and blue color levels.

SYNTAX CALL RECTFILC(XX1, YY1, XX2, YY2, HEIGHT, RED, GREEN BLUE)

<u>*ARGUMENTS*</u>

XX1,YY1	X,Y coordinates of lower left corner of rectangle.
XX2,YY2	X,Y coordinates of lower right corner of rectangle.
HEIGHT	Height of rectangle.
RED, GREEN, BLUE	RGB values of the fill color. Must lie between 0. and 1., inclusively.

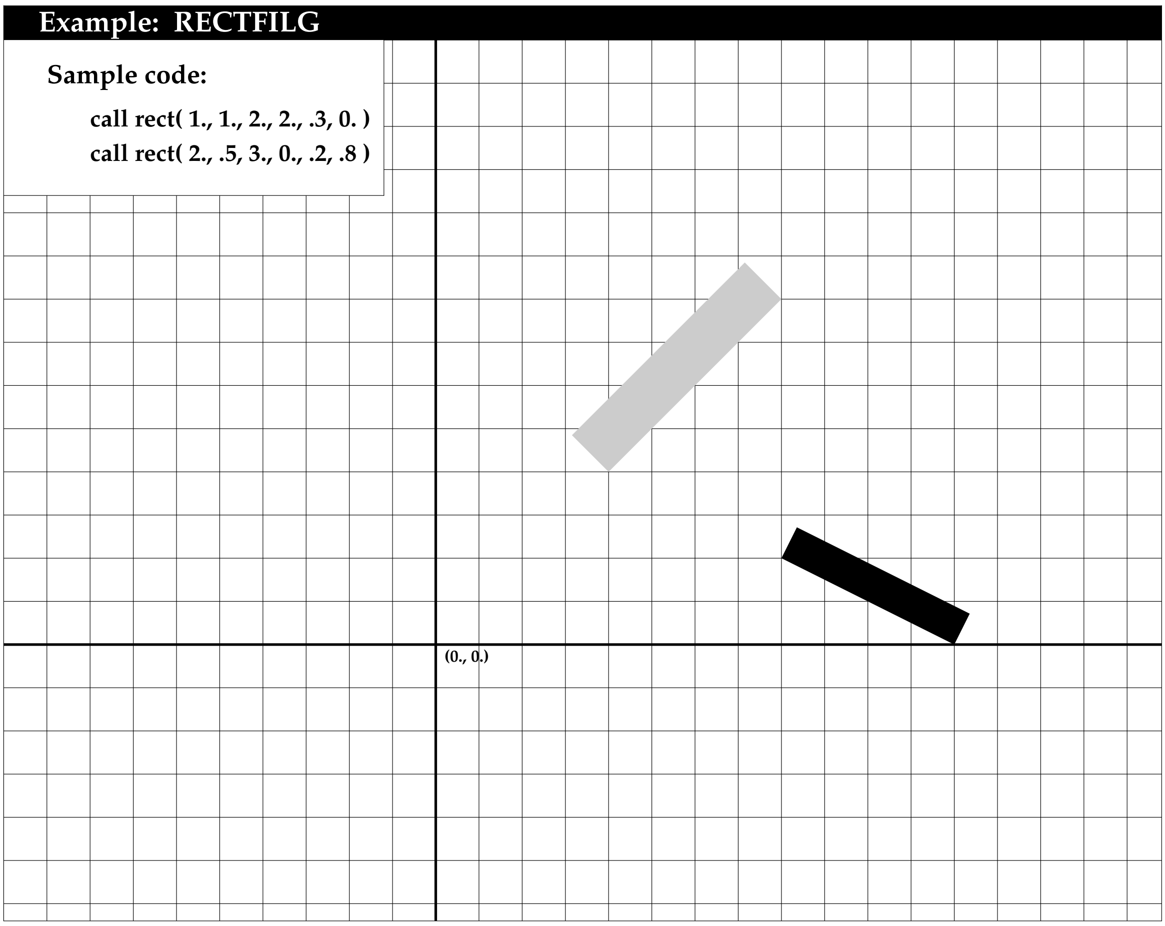
RECTFILG

PURPOSE **RECTFILG** draws a rectangle (or square) and fills it with a specified graylevel.

SYNTAX CALL RECTFILG (XX1, YY1, XX2, YY2, HEIGHT, GRYLEV)

ARGUMENTS

XX1,YY1	X,Y coordinates of lower left corner of rectangle.
XX2,YY2	X,Y coordinates of lower right corner of rectangle.
HEIGHT	Height of rectangle.
GRYLEV	Graylevel to use for filling rectangle. GRYLEV must lie beween 0. (black) and 1. (white), inclusive.



ROTATE

PURPOSE **ROTATE** rotates the current coordinate system by a specified angle. Essentially, ROTATE issues the PostScript *rotate* command.

SYNTAX CALL ROTATE (ANG)

ARGUMENTS

ANGLE Angle, measured counterclockwise, to rotate the current coordinate system.

RRECT

PURPOSE **RRECT** draws a rectangle (or square).with rounded corners. It can then fill the rectangle with the current graylevel or RGB values.

SYNTAX CALL RECT (XX1, YY1, WIDTH, HEIGHT, RAD, ANG, FILL)

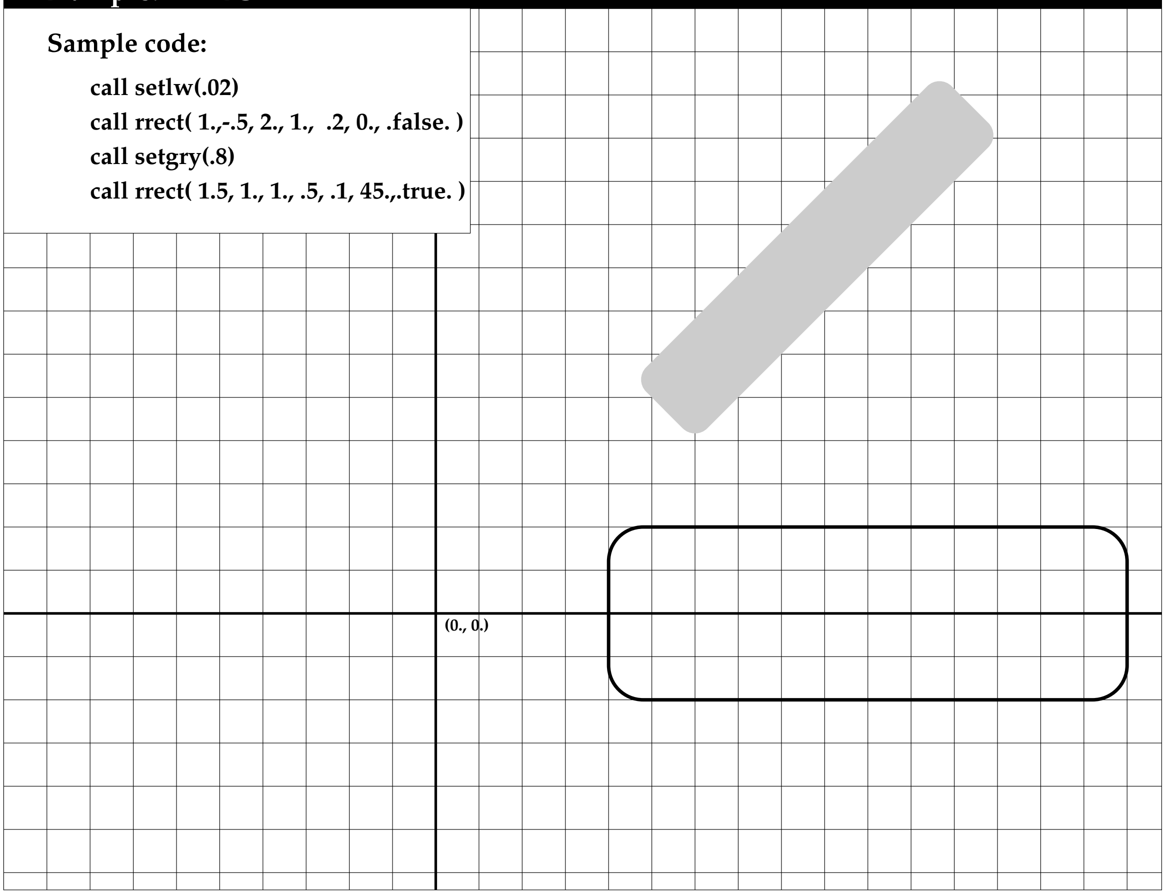
<u>*ARGUMENTS*</u>

XX1,YY1	X,Y coordinates of lower left corner of rectangle.
WIDTH	Width of rectangle.
HEIGHT	Height of rectangle.
RAD	Radius of rectangle corner curves.
ANG	Angle of rectangle rotation.
FILL	Logical value. If FILL=.true., then the rectangle is filled with the current graylevel or RGB values; otherwise, the rectangle is outlined.

Example: RRECT

Sample code:

```
call setlw(.02)
call rrect( 1.,-.5, 2., 1.,  .2, 0., .false. )
call setgry(.8)
call rrect( 1.5, 1., 1., .5, .1, 45.,.true. )
```



SETCOLR

PURPOSE **SETCOLR** sets the current red, green, and blue color levels.

SYNTAX CALL SETCOLR (RED, GREEN, BLUE)

ARGUMENTS

RED	Red color saturation level, between 0. and 1.0.
GREEN	Green color saturation level, between 0. and 1.0.
BLUE	Blue color saturation level, between 0. and 1.0.

SETFNT

PURPOSE **SETFNT** sets the current font.

SYNTAX CALL SETFNT (NFONT)

ARGUMENTS

NFONT Number of desired font between 1 and 35 (see table below).

The fonts sets 29 (Symbol) and 35 (ZapfDingbats) require the octal code of the character to be plotted. See the tables on the following pages for the character sets and corresponding octal code values for these two fonts.

Fonts for Subroutine SETFNT

1	AvantGarde-Book	*19*	*Helvetica-Oblique*
2	*AvantGarde-BookOblique*	20	Helvetica
3	**AvantGarde-Demi**	**21**	**NewCenturySchlbk-Bold**
4	***AvantGarde-DemiOblique***	***22***	***NewCenturySchlbk-BoldItalic***
5	**Bookman-Demi**	*23*	*NewCenturySchlbk-Italic*
6	***Bookman-DemiItalic***	24	NewCenturySchlbk-Roman
7	Bookman-Light	**25**	**Palatino-Bold**
8	*Bookman-LightItalic*	***26***	***Palatino-BoldItalic***
9	**Courier-Bold**	*27*	*Palatino-Italic*
10	***Courier-BoldOblique***	28	Palatino-Roman
11	*Courier-Oblique*	29	Symbol Σψμβολ
12	Courier	**30**	**Times-Bold**
13	**Helvetica-Bold**	***31***	***Times-BoldItalic***
14	***Helvetica-BoldOblique***	*32*	*Times-Italic*
15	**Helvetica-Narrow-Bold**	33	Times-Roman
16	***Helvetica-Narrow-BoldOblique***	*34*	*ZapfChancery-MediumItalic*
17	*Helvetica-Narrow-Oblique*	35	ZapfDingbats ❋❅❒❆⌘❇■❊✲❅▼▲
18	Helvetica-Narrow		

Characters and octal codes for Font Symbol

	!	∀	#	∃	%	&	∍	(	)	∗	+	,	−	.	/
40	41	42	43	44	45	46	47	50	51	52	53	54	55	56	57
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
60	61	62	63	64	65	66	67	70	71	72	73	74	75	76	77
≅	Α	Β	Χ	Δ	Ε	Φ	Γ	Η	Ι	ϑ	Κ	Λ	Μ	Ν	Ο
100	101	102	103	104	105	106	107	110	111	112	113	114	115	116	117
Π	Θ	Ρ	Σ	Τ	Υ	ς	Ω	Ξ	Ψ	Ζ	[	∴	]	⊥	_
120	121	122	123	124	125	126	127	130	131	132	133	134	135	136	137
‾	α	β	χ	δ	ε	ϕ	γ	η	ι	φ	κ	λ	μ	ν	ο
140	141	142	143	144	145	146	147	150	151	152	153	154	155	156	157
π	θ	ρ	σ	τ	υ	ϖ	ω	ξ	ψ	ζ	{	|	}	~	
160	161	162	163	164	165	166	167	170	171	172	173	174	175	176	177
€	ϒ	′	≤	⁄	∞	ƒ	♣	♦	♥	♠	↔	←	↑	→	↓
240	241	242	243	244	245	246	247	250	251	252	253	254	255	256	257
°	±	″	≥	×	∝	∂	•	÷	≠	≡	≈	…	⏐	⎯	↵
260	261	262	263	264	265	266	267	270	271	272	273	274	275	276	277
ℵ	ℑ	ℜ	℘	⊗	⊕	∅	∩	∪	⊃	⊇	⊄	⊂	⊆	∈	∉
300	301	302	303	304	305	306	307	310	311	312	313	314	315	316	317
∠	∇	®	©	™	∏	√	⋅	¬	∧	∨	⇔	⇐	⇑	⇒	⇓
320	321	322	323	324	325	326	327	330	331	332	333	334	335	336	337
◊	〈	®	©	™	∑	⎛	⎜	⎝	⎡	⎢	⎣	⎧	⎨	⎩	⎪
340	341	342	343	344	345	346	347	350	351	352	353	354	355	356	357
	〉	∫	⌠	⎮	⌡	⎞	⎟	⎠	⎤	⎥	⎦	⎫	⎬	⎭	
360	361	362	363	364	365	366	367	370	371	372	373	374	375	376	

Characters and octal codes for Font ZapfDingbats

	✁	✂	✃	✄	☎	✆	✇	✈	✉	☛	☞	✌	✍	✎	✏
40	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**
✐	✑	✒	✓	✔	✕	✖	✗	✘	✙	✚	✛	✜	✝	✞	✟
60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**
✠	✡	✢	✣	✤	✥	✦	✧	★	✩	✪	✫	✬	✭	✮	✯
100	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**
✰	✱	✲	✳	✴	✵	✶	✷	✸	✹	✺	✻	✼	✽	✾	✿
120	**121**	**122**	**123**	**124**	**125**	**126**	**127**	**130**	**131**	**132**	**133**	**134**	**135**	**136**	**137**
❀	❁	❂	❃	❄	❅	❆	❇	❈	❉	❊	❋	●	❍	■	❏
140	**141**	**142**	**143**	**144**	**145**	**146**	**147**	**150**	**151**	**152**	**153**	**154**	**155**	**156**	**157**
❐	❑	❒	▲	▼	◆	❖	◗	❘	❙	❚	❛	❜	❝	❞	
160	**161**	**162**	**163**	**164**	**165**	**166**	**167**	**170**	**171**	**172**	**173**	**174**	**175**	**176**	**177**
	❡	❢	❣	❤	❥	❦	❧	♣	♦	♥	♠	①	②	③	④
240	**241**	**242**	**243**	**244**	**245**	**246**	**247**	**250**	**251**	**252**	**253**	**254**	**255**	**256**	**257**
⑤	⑥	⑦	⑧	⑨	⑩	❶	❷	❸	❹	❺	❻	❼	❽	❾	❿
260	**261**	**262**	**263**	**264**	**265**	**266**	**267**	**270**	**271**	**272**	**273**	**274**	**275**	**276**	**277**
➀	➁	➂	➃	➄	➅	➆	➇	➈	➉	➊	➋	➌	➍	➎	➏
300	**301**	**302**	**303**	**304**	**305**	**306**	**307**	**310**	**311**	**312**	**313**	**314**	**315**	**316**	**317**
➐	➑	➒	➓	➔	→	↔	↕	➘	➙	➚	➛	➜	➝	➞	➟
320	**321**	**322**	**323**	**324**	**325**	**326**	**327**	**330**	**331**	**332**	**333**	**334**	**335**	**336**	**337**
➠	➡	➢	➣	➤	➥	➦	➧	➨	➩	➪	➫	➬	➭	➮	➯
340	**341**	**342**	**343**	**344**	**345**	**346**	**347**	**350**	**351**	**352**	**353**	**354**	**355**	**356**	**357**
	➱	➲	➳	➴	➵	➶	➷	➸	➹	➺	➻	➼	➽	➾	
360	**361**	**362**	**363**	**364**	**365**	**366**	**367**	**370**	**371**	**372**	**373**	**374**	**375**	**376**	

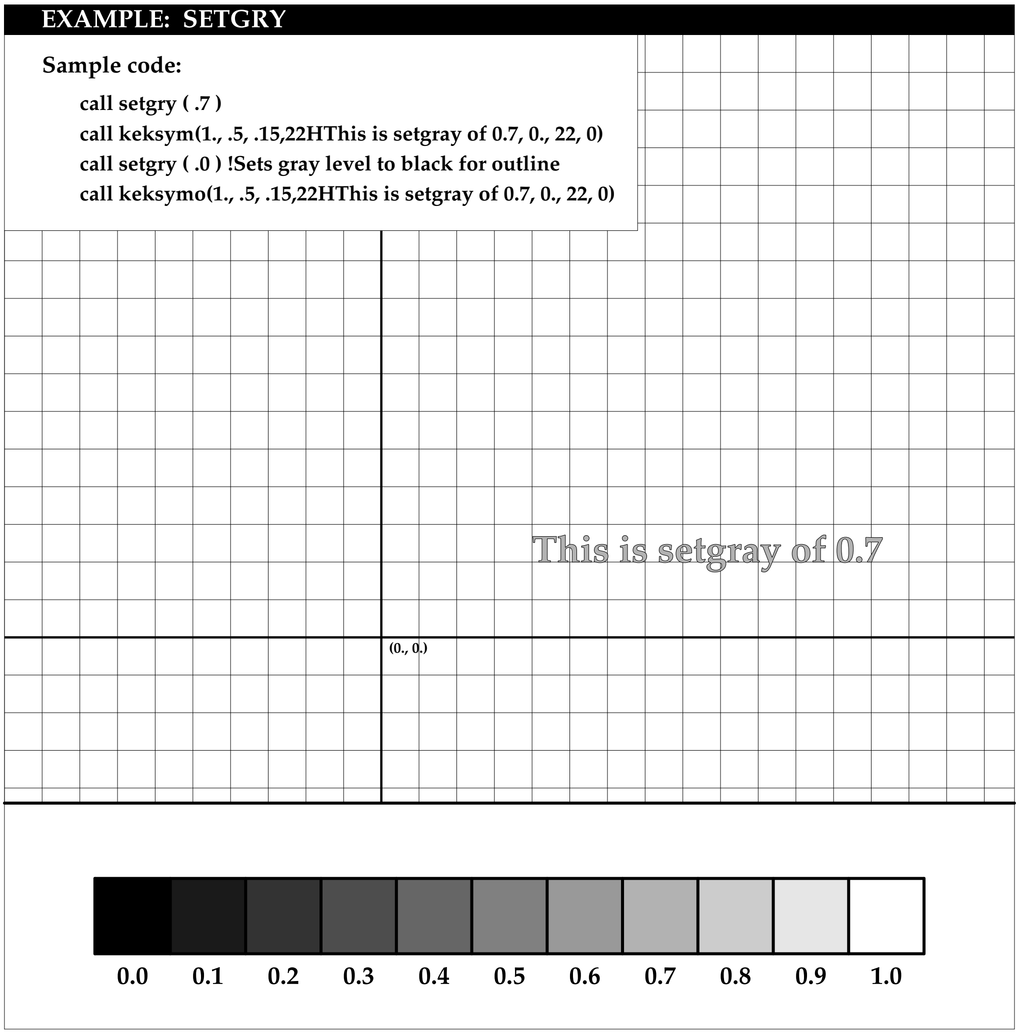
SETGRY

PURPOSE **SETGRY** sets the current gray level.

SYNTAX CALL SETGRY (GRYLVL)

<u>ARGUMENTS</u>

GRYLVL Gray level between 0. (black) and 1. (white).



SETLW

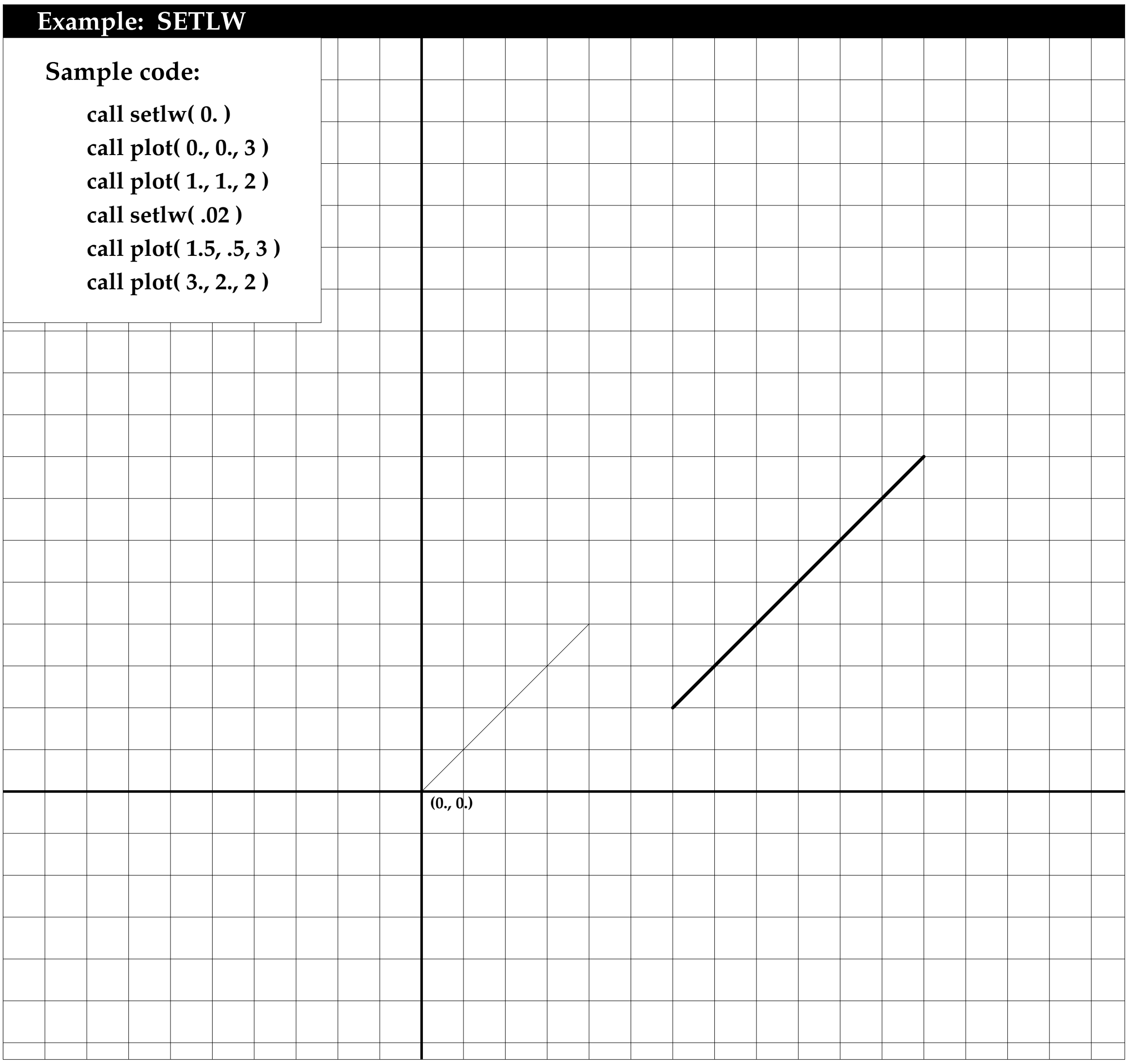
PURPOSE **SETLW** sets the current linewidth.

SYNTAX CALL SETLW (RLW)

<u>*ARGUMENTS*</u>

RLW	Line width.

After calling SETLW, RLW remains the current linewidth until SETLW is called again.



SIGMA

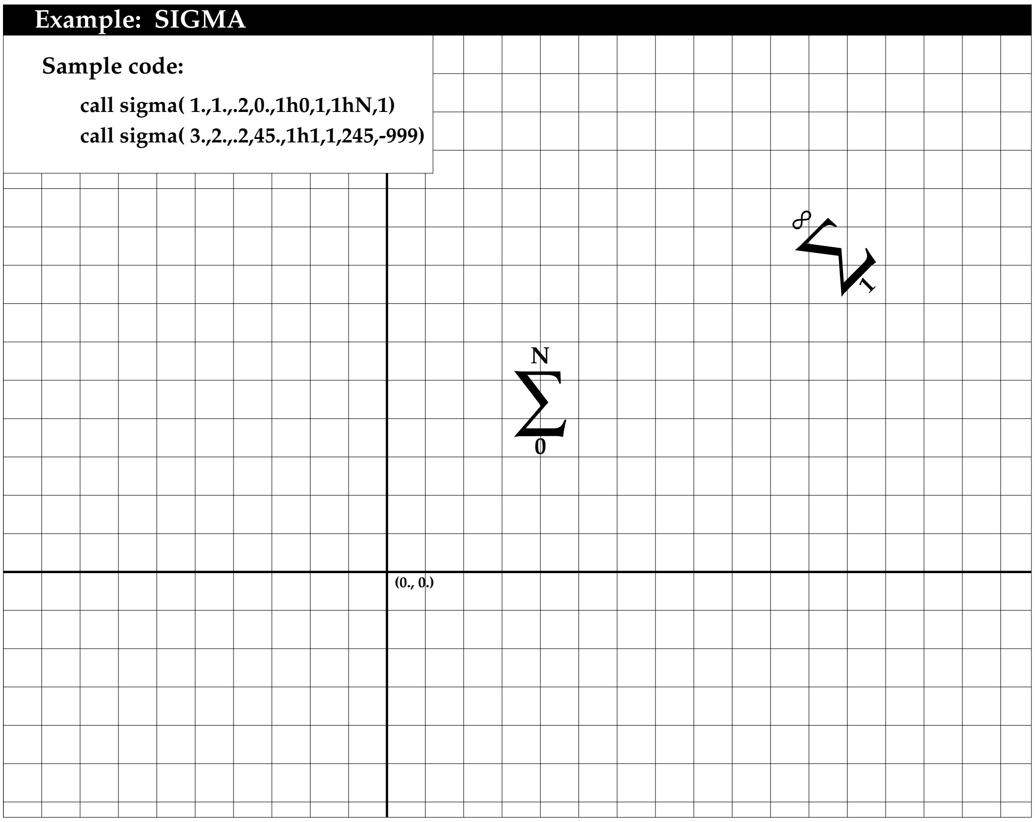
PURPOSE **SIGMA** draws the symbol Σ, with upper and lower limits.

SYNTAX CALL SIGMA (XPP, YPP, HEIGHT, ANG, LOWER, NL, LUPPER, NU)

<u>*ARGUMENTS*</u>

XPP, YPP	X, Y coordinates of the lower left edge of the symbol.
HEIGHT	Height of the symbol.
ANG	Angle, measured counterclockwise from the X-axis, at which the symbol is to be plotted.
LOWER	Character string specifying the lower limit (Hollerith).
NL	Number of characters in the lower limit character string.
LUPPER	Character string specifying the upper limit (Hollerith).
NU	Number of characters in the upper limit character string.

If NL (NU) is set to -999, then LOWER (LUPPER) is the octal code of the character in the font SYMBOL.



SLDCRV

PURPOSE **SLDCRV** connects points in two data arrays using a line with a specified thickness.

SYNTAX CALL SLDCRV (XARR, YARR, NPTS, THK)

ARGUMENTS

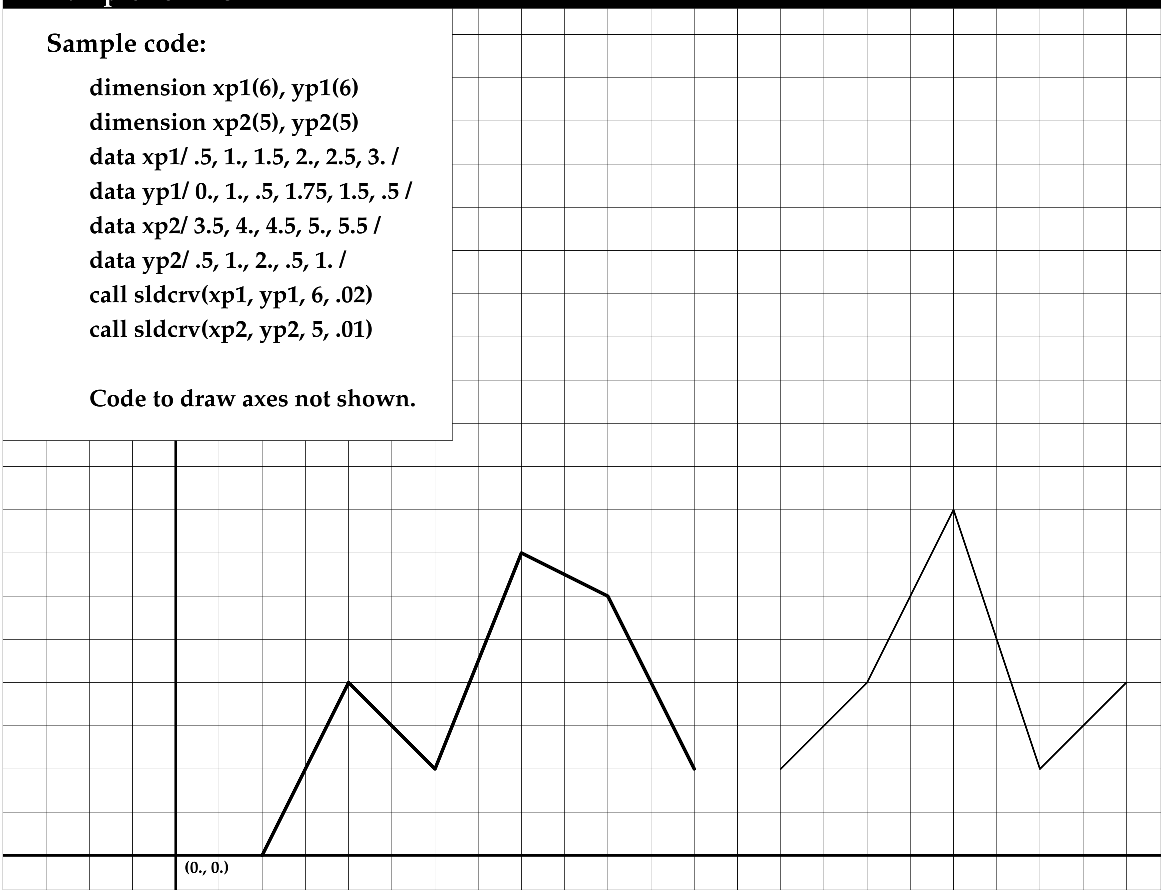
XARR	Array containing the X coordinates of the data points to be plotted.
YARR	Array containing the Y coordinates of the data points to be plotted.
NPTS	Number of points in the curve.
THK	Specifies the thickness of the curve. If THK = 0., the current linewidth is used.

Example: SLDCRV

Sample code:

```
dimension xp1(6), yp1(6)
dimension xp2(5), yp2(5)
data xp1/ .5, 1., 1.5, 2., 2.5, 3. /
data yp1/ 0., 1., .5, 1.75, 1.5, .5 /
data xp2/ 3.5, 4., 4.5, 5., 5.5 /
data yp2/ .5, 1., 2., .5, 1. /
call sldcrv(xp1, yp1, 6, .02)
call sldcrv(xp2, yp2, 5, .01)
```

Code to draw axes not shown.



SLDLIN

PURPOSE **SLDLIN** connects two points using a line with a specified thickness.

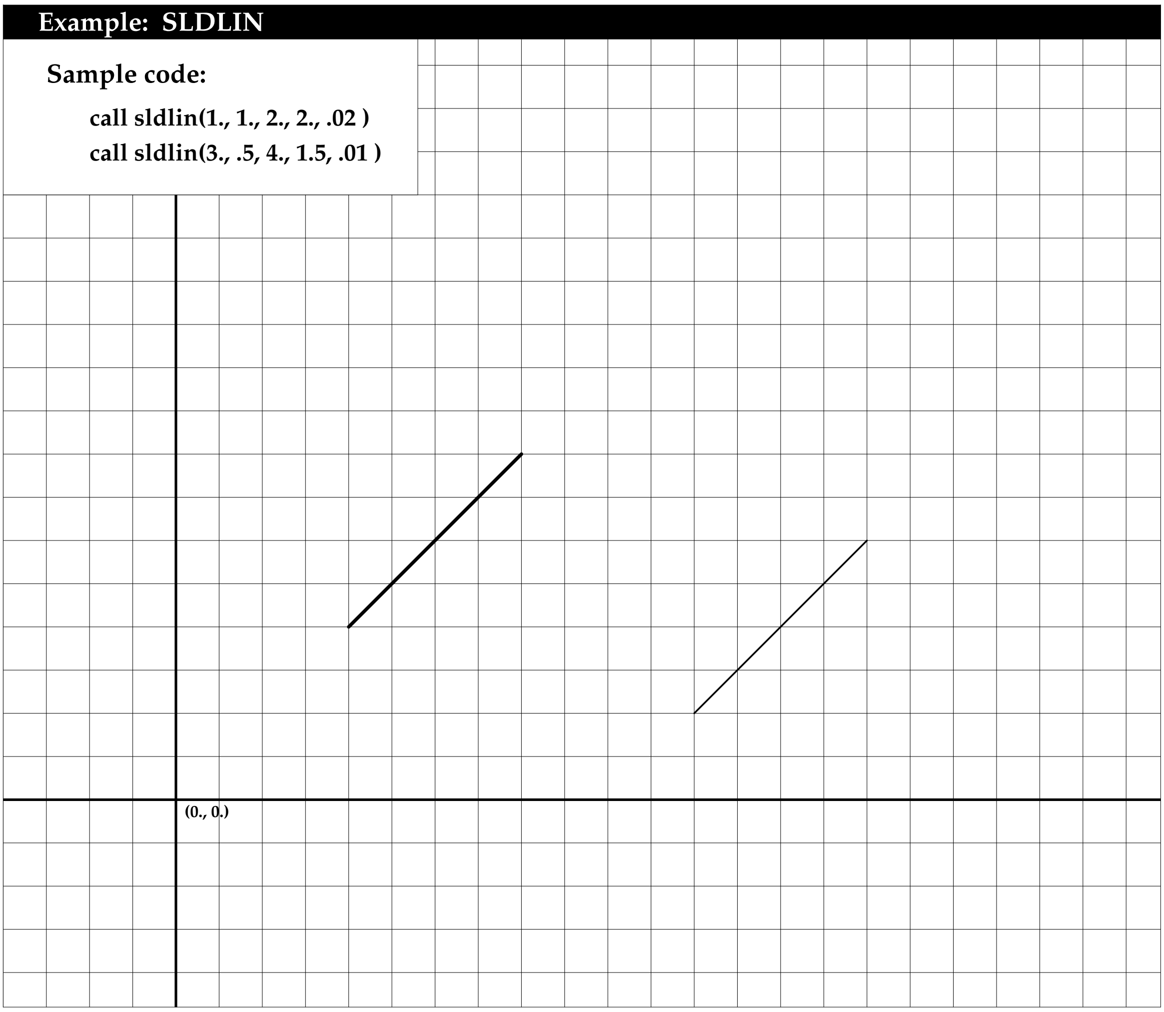
SYNTAX CALL SLDLIN (X1, Y1, X2, Y2, THK)

<u>*ARGUMENTS*</u>

X1,Y1 X,Y coordinates of first data point.

X2,Y2 X,Y coordinates of second data point.

THK Specifies the thickness of the connecting line. If THK=0., the current linewidth is used.



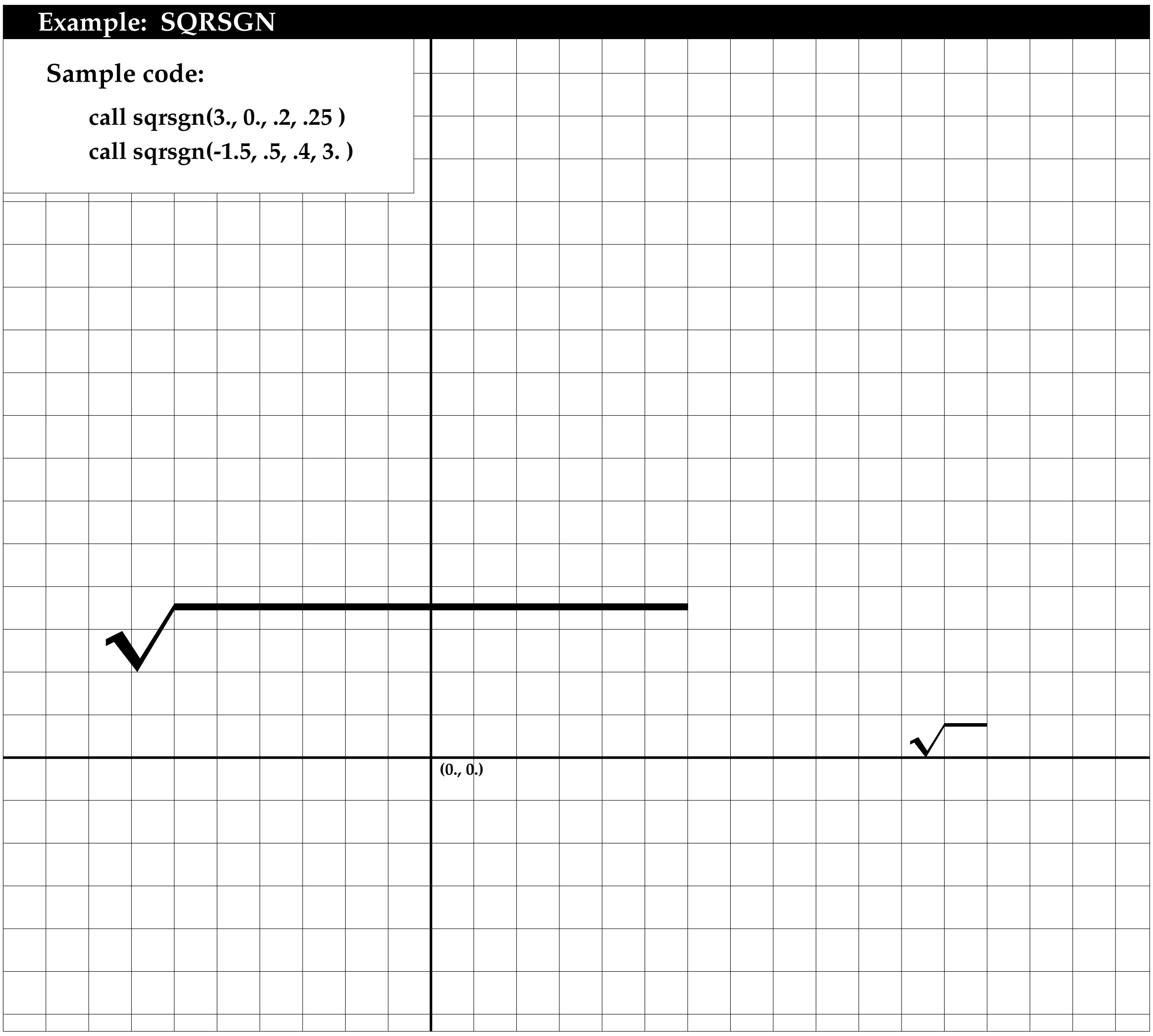
SQRSGN

PURPOSE **SQRSGN** draws a radical (square root) sign.

SYNTAX CALL SQRSGN (XPP, YPP, HEIGHT, RLEN)

<u>*ARGUMENTS*</u>

XPP,YPP	X,Y coordinates of lower left corner of the radical.
HEIGHT	Height of the radical.
RLEN	Length of the top of the radical.



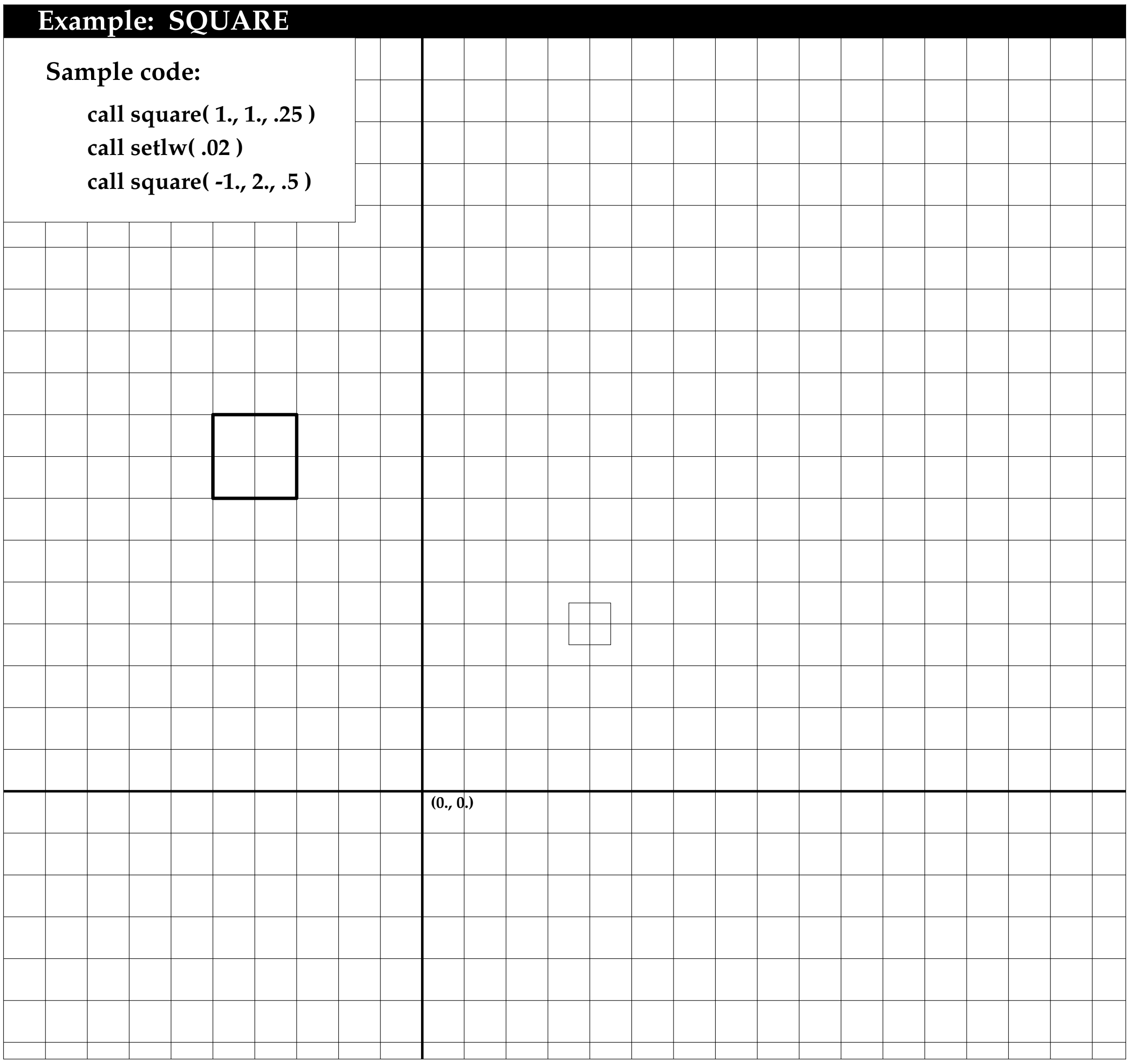
SQUARE

PURPOSE **SQUARE** draws a centered square.

SYNTAX CALL SQUARE (XC, YC, SIDE)

ARGUMENTS

XC,YC	X,Y coordinates of the center of the square.
SIDE	Length of the sides of the square.



STROKE

PURPOSE **STROKE** calls the PostScript operator *stroke* (which paints or draws the current path).

SYNTAX CALL STROKE

ARGUMENTS

None

SUBBER

PURPOSE **SUBBER** draws a subscript.

SYNTAX CALL SUBBER (ISUB, NSUB, SIZE, ANG)

ARGUMENTS

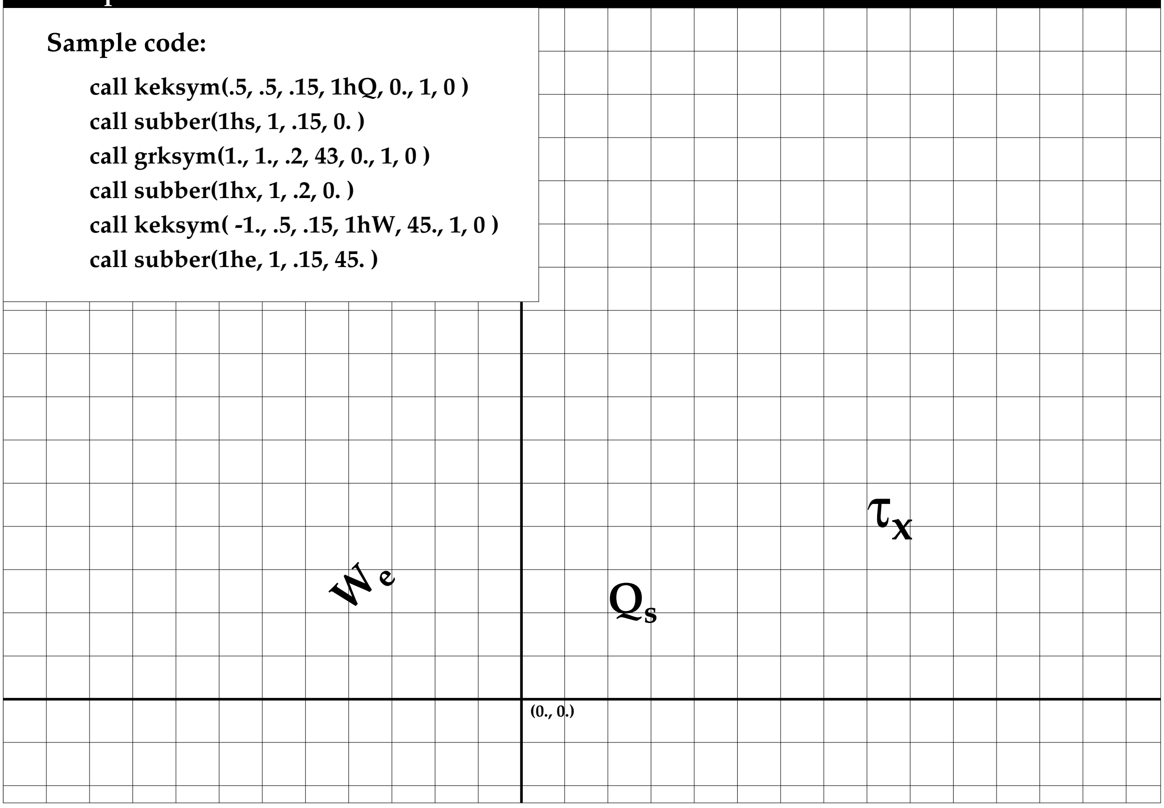
ISUB	Hollerith variable specifying the subscript character string.
NSUB	Number of characters in the subscript.
SIZE	Height of the **variable to be subscripted**. It is **not** the height of the subscript characters themselves.
ANG	Angle of the subscripted variable.

Note: SUBBER must be called immediately after the call to create the subscripted variable.

Example: SUBBER

Sample code:

```
call keksym(.5, .5, .15, 1hQ, 0., 1, 0 )
call subber(1hs, 1, .15, 0. )
call grksym(1., 1., .2, 43, 0., 1, 0 )
call subber(1hx, 1, .2, 0. )
call keksym( -1., .5, .15, 1hW, 45., 1, 0 )
call subber(1he, 1, .15, 45. )
```



SUBBERSP

PURPOSE **SUBBERSP** draws a subscript of "special" characters.

SYNTAX CALL SUBBERSP (NSET, NFNT, ITITLE, NCHR, HEIGHT, ANG

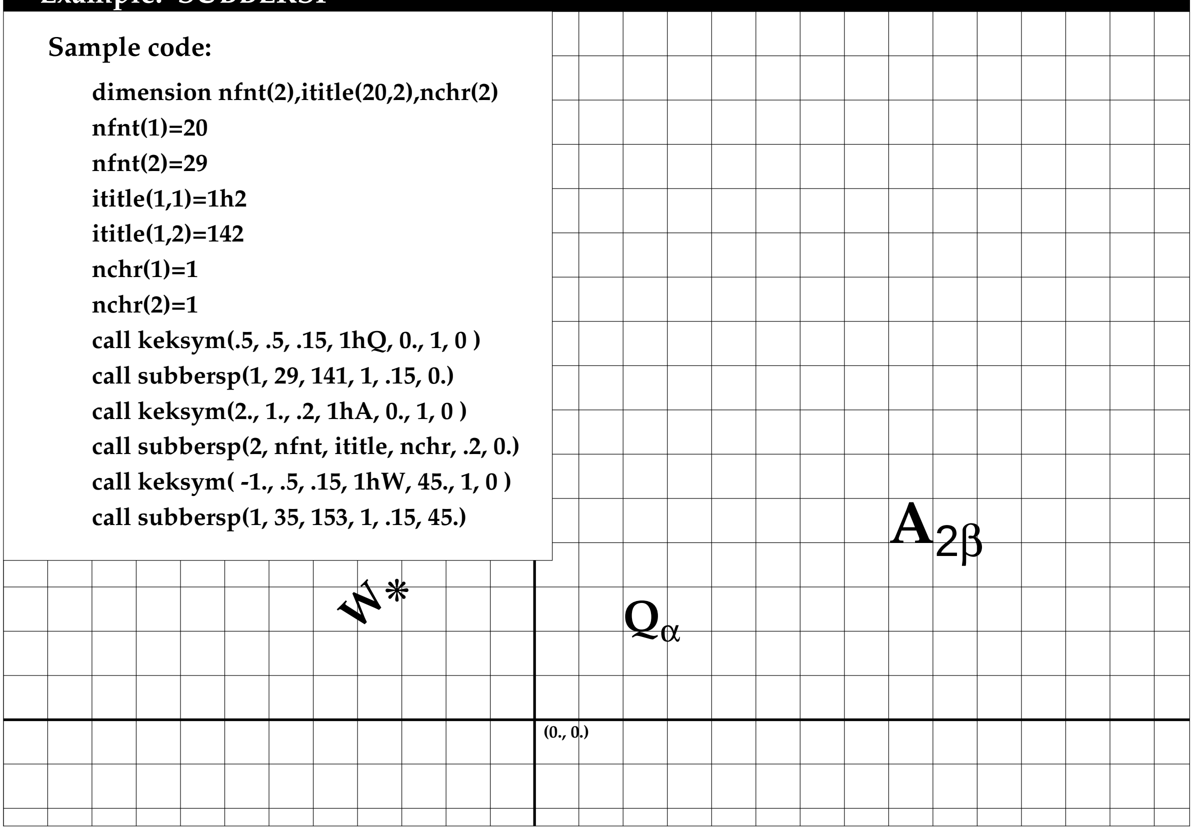
<u>*ARGUMENTS*</u>

NSET	Number of different font sets needed.
NFNT	Array holding the font numbers 1 to NSET.
ITITLE	ITITLE holds the octal codes for characters of fonts 29 or 35; otherwise, it hold the text characters themselves.
NCHR	Number of characters of a given font. Usually one, but can be greater than one for fonts other than 29 and 35.
HEIGHT	Height of the **variable to be subscripted**. It is **not** the height of the subscript characters themselves.
ANG	Angle of the subscripted variable.
Note:	SUBBERSP must be called immediately after the call to create the subscripted variable.

Example: SUBBERSP

Sample code:

```
dimension nfnt(2),ititle(20,2),nchr(2)
nfnt(1)=20
nfnt(2)=29
ititle(1,1)=1h2
ititle(1,2)=142
nchr(1)=1
nchr(2)=1
call keksym(.5, .5, .15, 1hQ, 0., 1, 0 )
call subbersp(1, 29, 141, 1, .15, 0.)
call keksym(2., 1., .2, 1hA, 0., 1, 0 )
call subbersp(2, nfnt, ititle, nchr, .2, 0.)
call keksym( -1., .5, .15, 1hW, 45., 1, 0 )
call subbersp(1, 35, 153, 1, .15, 45.)
```



SUBSUP

PURPOSE **SUBSUP** draws a subscript and superscript.

SYNTAX CALL SUBSUP (ISUB, NSUB, ISUP, NSUP, SIZE, ANG)

ARGUMENTS

ISUB Hollerith variable specifying the subscript character string.

NSUB Number of characters in the subscript.

ISUP Hollerith variable specifying the superscript character string.

NSUP Number of characters in the superscript.

SIZE Height of the **variable to be subscripted and superscripted**. It is **not** the height of the subscript and superscript characters themselves.

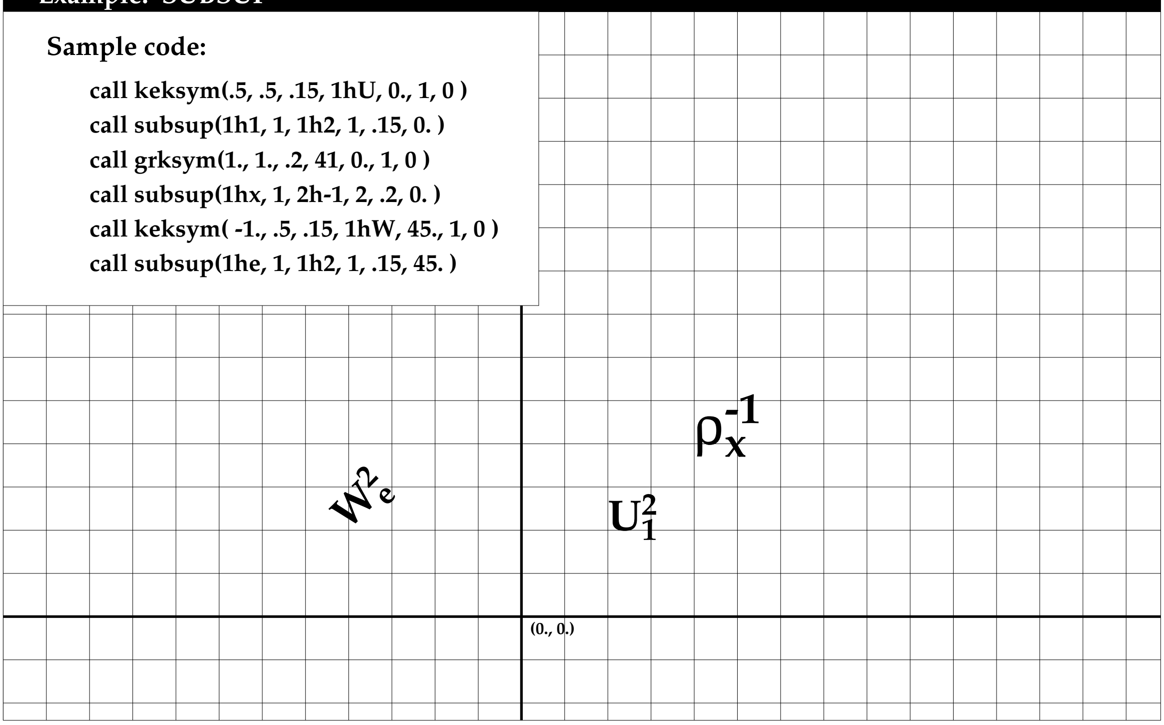
ANG Angle of the subscripted and superscripted variable.

Note: SUBSUP must be called immediately after the call to create the subscripted and superscripted variable.

Example: SUBSUP

Sample code:

```
call keksym(.5, .5, .15, 1hU, 0., 1, 0 )
call subsup(1h1, 1, 1h2, 1, .15, 0. )
call grksym(1., 1., .2, 41, 0., 1, 0 )
call subsup(1hx, 1, 2h-1, 2, .2, 0. )
call keksym( -1., .5, .15, 1hW, 45., 1, 0 )
call subsup(1he, 1, 1h2, 1, .15, 45. )
```



SUBSUPSP

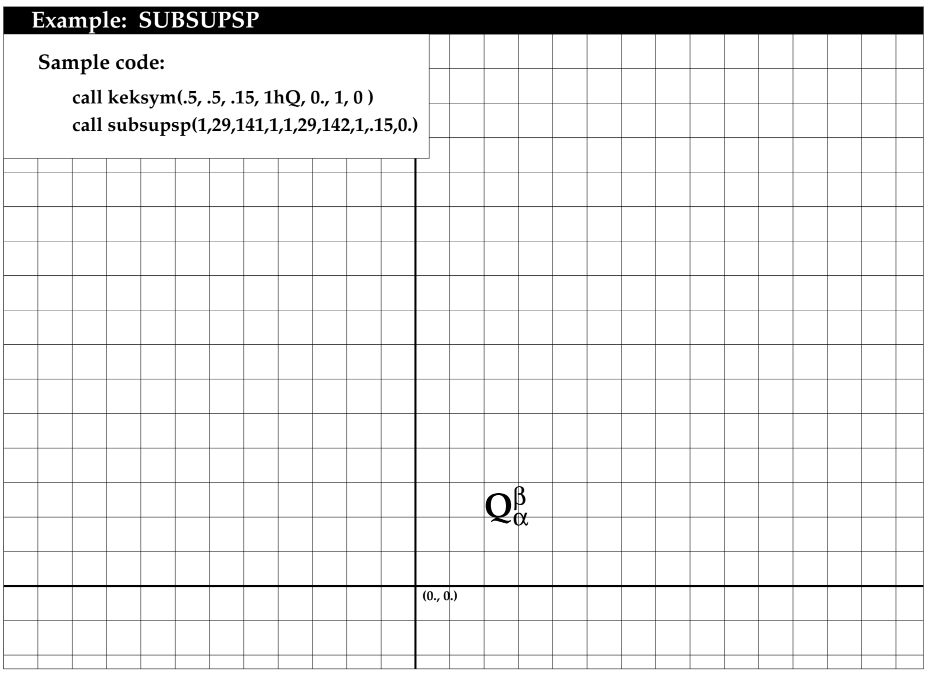
PURPOSE **SUBSUPSP** draws subscripts and superscripts of "special" characters.

SYNTAX CALL SUBSUPSP (NSUB, NFNTSB, ITITLESB, NCHRSB, NSUP, NFNTSP,ITITLESP, NCHRSP, HEIGHT, ANG)

ARGUMENTS

NSUB, NSUP	Number of different font sets needed for subscripts and superscripts, respectively.
NFNTSB, NFNTSP	Arrays holding the font numbers 1 to NSET for subscripts and superscripts, respectively.
ITITLESB, ITITLESP	Arrays holding the octal codes for characters of fonts 29 or 35; otherwise, holding text characters themselves.
NCHRSB, NCHRSP	Number of characters of a given font for subscripts and superscripts, respectively. Usually one, but can be greater than one for fonts other than 29 and 35.
HEIGHT	Height of the **variable to be subscripted**. It is **not** the height of the subscript characters themselves.
ANG	Angle of the subscripted variable.

Note: SUBSUPSP must be called immediately after the call to create the subscripted variable.



SUPER

PURPOSE **SUPER** draws a superscript.

SYNTAX CALL SUPER (ISUP, NSUP, SIZE, ANG)

<u>ARGUMENTS</u>

ISUP Hollerith variable specifying the superscript character string.

NSUP Number of characters in the superscript.

SIZE Height of the **variable to be superscripted**. It is **not** the height of the subscript characters themselves.

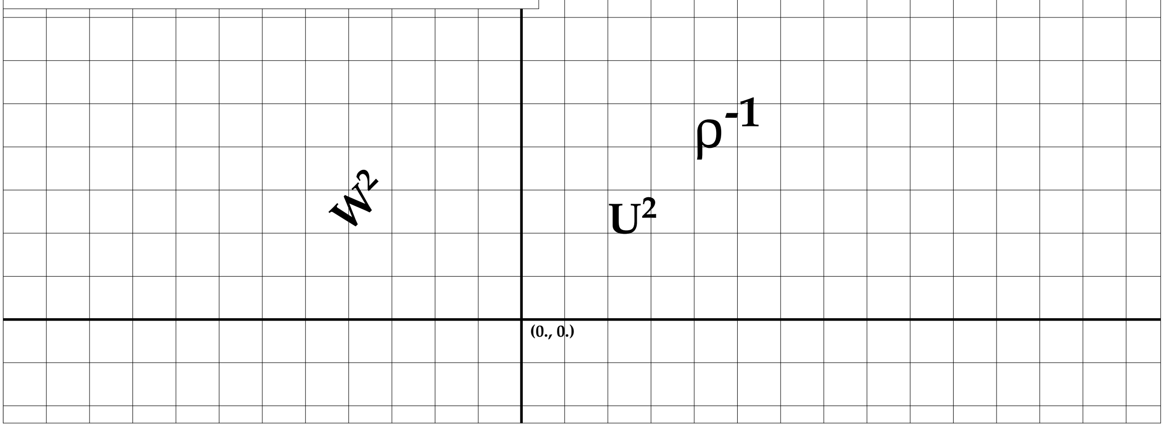
ANG Angle of the superscripted variable.

Note: SUPER must be called immediately after the call to create the superscripted variable.

Example: SUPER

Sample code:

```
call keksym(.5, .5, .15, 1hu, 0., 1, 0 )
call super(1h2, 1, .15, 0. )
call grksym(1., 1., .2, 41, 0., 1, 0 )
call super( 2h-1, 2, .2, 0. )
call keksym( -1., .5, .15, 1hW, 45., 1, 0 )
call super(1h2, 1, .15, 45. )
```



SUPERSP

PURPOSE **SUPERSP** draws a superscript of "special" characters.

SYNTAX CALL SUPERSP (NSET, NFNT, ITITLE, NCHR, HEIGHT, ANG)

ARGUMENTS

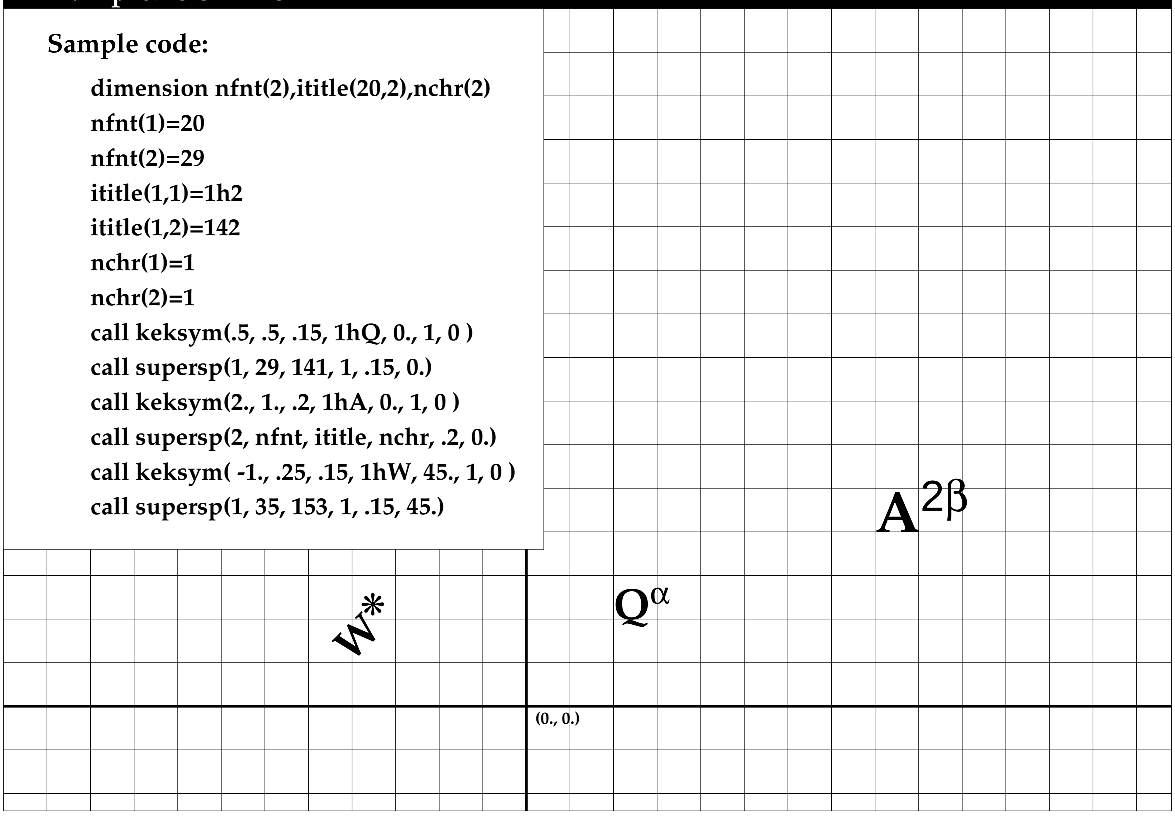
NSET	Number of different font sets needed.
NFNT	Array holding the font numbers 1 to NSET.
ITITLE	ITITLE holds the octal codes for characters of fonts 29 or 35; otherwise, it hold the text characters themselves.
NCHR	Number of characters of a given font. Usually one, but can be greater than one for fonts other than 29 and 35.
HEIGHT	Height of the **variable to be superscripted**. It is **not** the height of the superscript characters themselves.
ANG	Angle of the superscripted variable.

Note: SUPERSP must be called immediately after the call to create the superscripted variable.

Example: SUPERSP

Sample code:

```
dimension nfnt(2),ititle(20,2),nchr(2)
nfnt(1)=20
nfnt(2)=29
ititle(1,1)=1h2
ititle(1,2)=142
nchr(1)=1
nchr(2)=1
call keksym(.5, .5, .15, 1hQ, 0., 1, 0 )
call supersp(1, 29, 141, 1, .15, 0.)
call keksym(2., 1., .2, 1hA, 0., 1, 0 )
call supersp(2, nfnt, ititle, nchr, .2, 0.)
call keksym( -1., .25, .15, 1hW, 45., 1, 0 )
call supersp(1, 35, 153, 1, .15, 45.)
```



SYMBOL

PURPOSE **SYMBOL** plots a character string. It is similar to subroutine KEKSYM, except that justification and continuation are not supported.

SYNTAX CALL SYMBOL (XPP, YPP, HEIGHT, IBCD, ANG, NCHAR)

ARGUMENTS

XPP,YPP X,Y coordinates of the lower left corner of the first character of the string to be plotted.

HEIGHT Height of character string to be plotted.

IBCD Character string to be plotted (Hollerith).

ANG Angle, measured counterclockwise from the X-axis, at which the character string is to be plotted.

NCHAR Number of characters in the string.

Example: SYMBOL

Sample code:

```
call symbol( -1.5, .5, .15, 8hVelocity, 0., 8 )
call symbol(1.5,.5, .15, 8hDistance, 45., 8 )
```